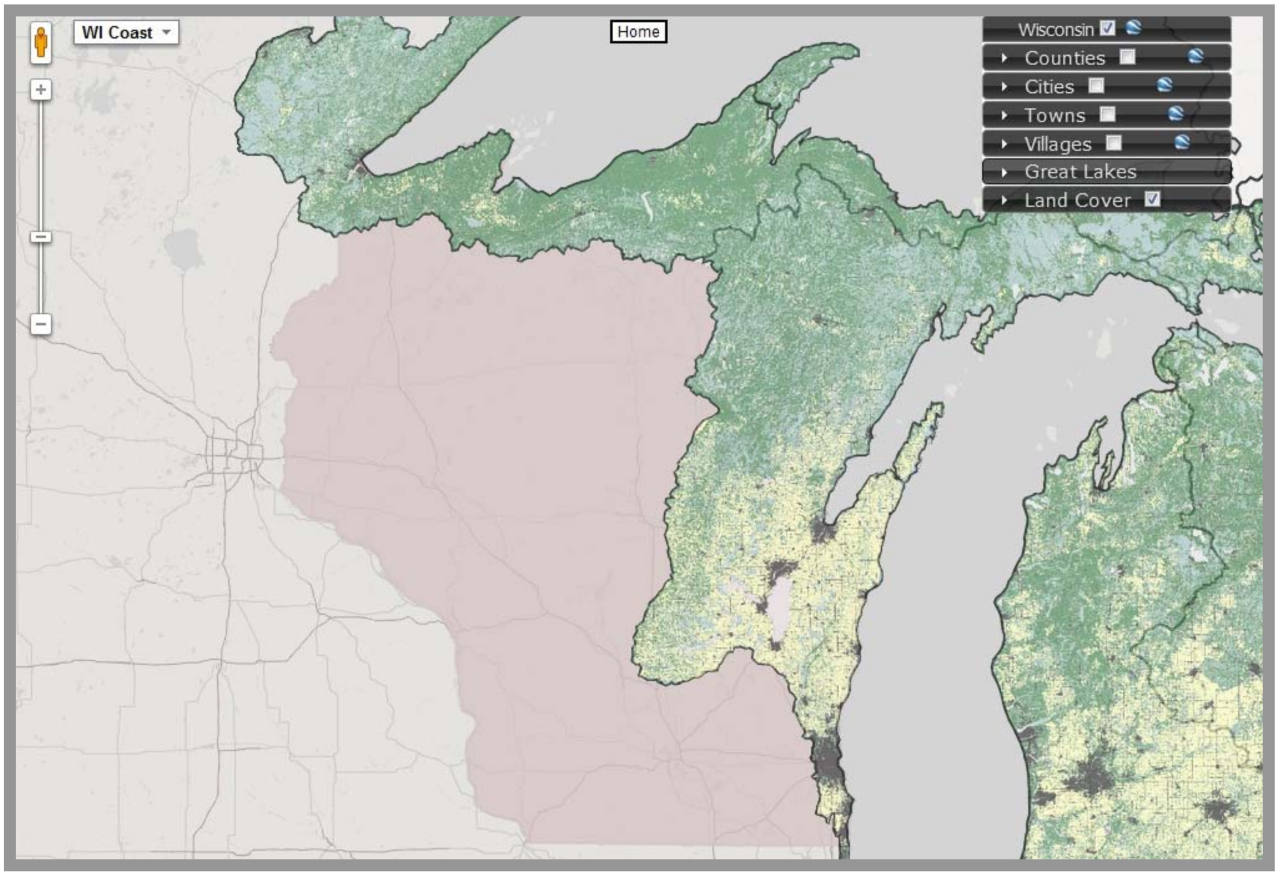
wisconsin coastal atlas white paper



User-Centered Design for Coastal Web Atlas Maps and Tools: A Process Manual

Carl Sack
UW Sea Grant Institute

August, 2013

Sea Grant
University of Wisconsin

Introduction

Why User-Centered Design?

This report focuses on the process of User-Centered Design (UCD) as it may be employed in the creation of coastal web atlas mapping applications and tools. UCD is an iterative, multi-stage software development process that involves user input at each stage of development. This user input is critical to the success of an application, and has been shown to increase the productivity of users, reduce user errors, reduce the time needed for training and system support, improve the acceptance and adoption of the system, and enhance the reputation of the system designer (Maguire 2001).

A coastal web atlas is a means of organizing, presenting, and sharing spatial data for the coast. It functions as a platform for interactive maps and decision support tools for coastal management, and is part of a growing coastal spatial data infrastructure. According to O'Dea et al. (2007):

> "[T]he design and usability of an atlas are keys to its success. An atlas should clearly communicate its purpose, be visually appealing, be kept as simple as possible, use efficient technology and management systems and have a flexible design to enable growth and change over time. Ultimately its success relies on the atlas users, so efforts should be made regularly to ensure that it meets the needs of those users" (33).

User-Centered Design adopts this philosophy of design and provides a framework for application development addressed toward the goal of maximizing usability and user satisfaction (Haklay 2010).

The UCD Process

The key principles of UCD are an early focus on users and tasks, empirical measurements, and iterative design (Gould and Lewis 1985). Various researchers (e.g., Medyckyj-Scott and Hearnshaw 1993, Gabbard et al. 1999, Maguire 2001, Slocum et al. 2003, Robinson et al. 2005, Haklay 2010) have divided the UCD process into between three and eight iterative stages, with alternating development work and user feedback at each stage. The ISO (1999) Standard 13407, "Human Centered Design for Interactive Systems," specifies five major stages: user-centered planning, determining the context of use, specifying user and organizational requirements, producing design solutions, and evaluating the design against the established requirements.

The version of the process described in the following sections largely mirrors the process used by Robinson et al. (2005) as modified by Sack (2013), which is visualized in Figure 1. The latter study involved the development of a web mapping application for crowdsourced information, or *wikimap.* It will be used as a case study for the remainder of this paper. Each stage of the UCD process will be covered in its own section. These stages are:

1) *Needs Assessment:* Also labeled *requirements specification* (Medyckyj-Scott and Hearnshaw 1993)*, user task analysis* (Gabbard et al. 1999, Slocum et al. 2003), and *work domain analysis* (Robinson et al. 2005), this stage involves consulting stakeholders or potential users to determine the target users, information and data sources, goals and objectives, use contexts, intended tasks and interactions, and technical requirements of

the system to be developed.

2) *Concept Design:* This stage involves the creation of a detailed list of system specifications using the information gathered during the Needs Assessment. These may include the semantic ontologies, interface features, required software and hardware, design constraints, standards, and usability targets of the system. The Needs Assessment and Concept Design components together can be compared to the first three stages of ISO 13407.
3) *Prototyping:* The iterative development of multiple visual mock-ups and partially-working system components, with feedback on each successive prototype, demonstrates the look and feel of the system to potential users before the developer is invested in its completed development. This process can catch problems while they are still minor, saving considerable time and effort. It is comparable to the fourth stage of ISO 13407.
4) *Formative Evaluation:* An initial round of formal usability studies can screen an alpha-version of the system for problems and elicit suggestions from more potential users than may have been reached during early stages.
5) *Debugging and Release:* The two components of this stage may occur in either order or concurrently. Release involves the announcement of the availability of the system for public use and subsequent promotion of the application. While most debugging should take place before the release, in reality it is an ongoing process that continues in the form of application support as long as the application is live.
6) *Summative Evaluation:* A final round of usability studies can draw conclusions as to whether the application successfully met the utility and usability targets specified in the Concept Design and generally agreed-upon design standards and conventions. This stage is comparable to the final stage of ISO 13407.

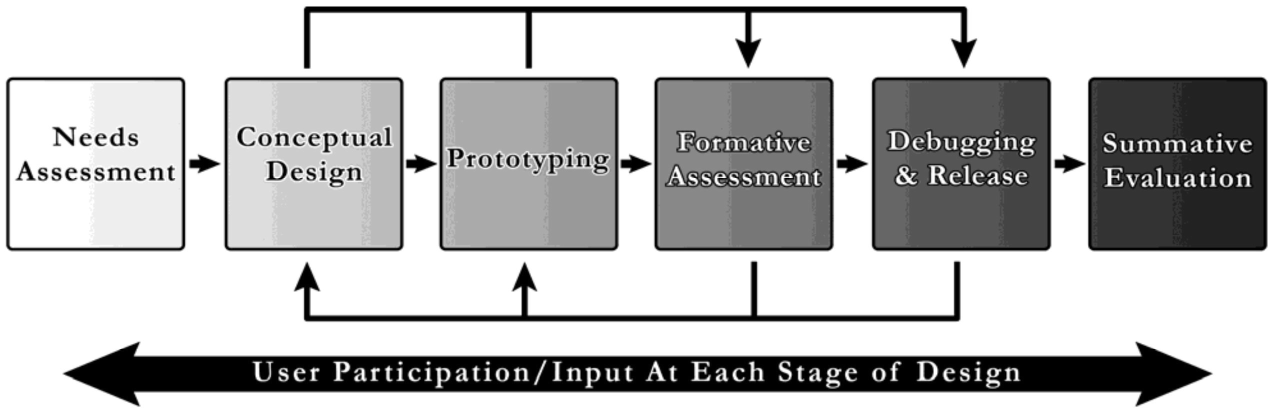


Figure 1: The User-Centered Design process, as applied by Sack (2013) (after Robinson et al. 2005).

Stage 1: Needs Assessment

The initial stage of the UCD process is a formal, qualitative study of the purpose, goals, objectives, target users, user needs, intended tasks and interactions, and technical requirements of the application (Medyckyj-Scott and Hearnshaw 1993). In keeping with the first two principles of UCD, this stage should involve consultation with potential application users right off the bat.

Participants

Here the term 'user' must be defined. Various practitioners interpret the term to mean either a *typical end user* of the system or a domain expert *stakeholder* with a vested interest in the final outcome for a very specific purpose (Roth and Harrower 2008). In specialized GIS and spatial decision support tools, there may be little distinction between these two categories. However, web mapping applications available to the public attract a much more diverse and less predictable set of end users than traditional GIS used by specialists (Tsou and Curran 2008) . This presents a problem for truly *user*-centered design: how does the developer decide who the typical users *are* and recruit them accordingly?

In the UCD process employed by Sack (2013), a group of domain experts with little knowledge of Cartography, GIS, or web development was recruited for the Needs Assessment. This selection process had the advantage of involving a set of potential end users with strong ideas about what would constitute usability for such an application. However, this initial group was not necessarily the same set of users as those involved in the formative or summative usability testing. Later measurements involved progressively wider sets of end users.

In a coastal atlas context, reaching stakeholders requires developing partnerships with local agencies, community organizations, leaders, and other interested parties prior to the Needs Assessment. The level of commitment of community partners can determine the quality of the responses received and the willingness of stakeholders to remain engaged throughout the iterative design process. A *stakeholder analysis*, or detailed research into the age, gender, cultural, and other demographic information of the community in question, can help improve the developer's understanding of community needs. Developers should identify community leaders who can be encouraged to adopt the project and introduce it to other stakeholders. Transparency and informed consent are crucial ethical components of stakeholder participation. Many academic institutions will require institutional review of the project prior to any formal studies (NOAA 2009).

Methods

Several qualitative survey methods can be employed to fulfill a Needs Assessment. These may include interviews, focus groups, surveys, questionnaires, Delphi method, and card sorting (Roth 2011b). Each is discussed below. Research into prior studies of similar scenarios can guide the structure and design of the study. The method(s) should be chosen based on available time, funding, availability of stakeholders, and the nature of the desired application (Hacklay 2010).

Interviews are the most tried-and-true method of gaining user feedback on the initial application design. An interview is a one-on-one conversation between a developer or researcher and a user in which the developer asks a series of structured and/or unstructured questions (Roth 2011b). Structured interviews involve the use of a script with very specific questions that the interviewer does not deviate from. While structured interviews can be useful for scientific studies, semi-structured or unstructured interviews are more practical for UCD, as they allow space for a two-way dialogue to generate a thorough understanding of the participant's thoughts and opinions. An unstructured interview is a free-flowing conversation, while a semi-structured interview starts with a specified set of preordered questions but allows the interviewer discretion to probe potentially interesting responses (DiCicco-Bloom and Crabtree 2006). Interview questions should be probing but not leading or restrictive. The interview should be recorded, with the permission of the interviewee, using an unobtrusive audio device (Cairns and Cox 2008).

A *focus group* is a facilitated discussion among a group of users or stakeholders. This environment allows participants to stimulate each other's ideas, resulting in an emerging consensus that is greater than the collective views of the individuals (Maguire 2001). Good focus groups require 3-10 participants with a range of views to meet at the same place and time, and must have excellent facilitation to ensure that all voices are heard equally (Morgan and Scannell 1998).

The *Delphi method* is similar in outcome to a focus group, but relies on anonymous and distributed feedback from participants collected in several rounds, with each round summarized by the facilitator and the summary shared to prompt discussion in the next round. This method is asynchronous and does not require participants to be located in the same place, but takes much longer, with each round lasting up to a week (Roth 2011b).

Card sorting is a method of facilitation that may be used in an individual or small group context. Using either physical paper or specialized software, participants are given a set of ideas (or cards) and asked to group them according to similarity. This method can be a useful supplement for arranging desired information or features of the application (Robinson et al. 2005).

Questionnaires may also be used to gather information and opinions from target users. However, while they can potentially reach a broader range of participants with less time and effort, the feedback received will be not nearly as rich with insight as can be achieved by direct contact with a selected group of target users, so they should not be the sole method used in the Needs Assessment stage (Haklay 2010).

Parameters

It is important to know what questions to ask of target users during the Needs Assessment so as to shape as many of the design parameters as possible as thoroughly as possible. This requires thinking through what parameters will need to be covered, and how to structure the questioning to get the most thoughtful results. The design framework proposed by Tsou and Curran (2008), based on Garrett (2002), may be helpful in thinking through and ordering questions about the system design.

The Garrett framework, shown in Figure 2, consists of five levels of abstraction, or *planes*, ordered from most broad and abstract to most detailed and concrete: strategy, scope, structure, skeleton, and surface. At each plane, parameters are identified for two system components: information to be included and user interface tasks. *Strategy* includes the user needs that the system will address and the more specific objectives for the application. *Scope* identifies what features and content the site needs to fulfill user requirements. *Structure* identifies the interactions, or how the system will behave in response to the user, and the information architecture, or arrangement of content elements. *Skeleton* specifies how information will be visually presented to the user, including the interface design (arrangement of the interface elements) and navigation design (available tools that allow the user to move through the application's architecture). *Surface* is the overall look of the final product.

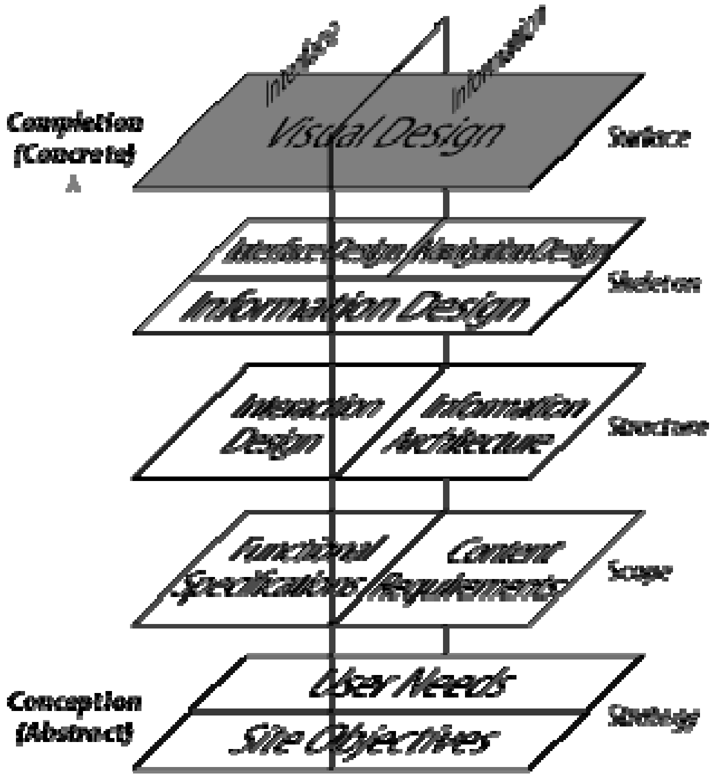


Figure 2: Garrett's (2002) five planes of software design.

A slightly different organization of parameters was used in the case study presented in Sack (2013). Participants were presented with sets of interview questions grouped thematically into six sections. Each was asked to characterize: 1) the potential users of the application, 2) what information layers it should contain, 3) the potential community impact of the application, 4) what levels of technology were appropriate for the mapping project, 5) what specific features or tools would be useful in the application, and 6) locations within the mapped area that fit a set of thematic categories developed based on secondary sources. Questions were grouped this way based on the context of the application, which was part of a broader participatory mapping project.

Coding Observations

All sessions of direct contact with participants should be audio and/or video recorded and transcribed for coding. Although time consuming, transcription should be exact (word-for-word) to ensure accuracy. *Coding* (assigning one-word codes to stand for specific concepts and themes) can be used to analyze participant responses. *Open coding* does not start with any pre-determined set of codes, but picks out concepts in each participant answer and joins similar concepts into higher-level categories. An example of open coding is presented in Figure 3. Two other forms of coding, *axial coding* and *selective coding,* are more useful for participant observation where high-level phenomena can be identified first, and then participant conditions and strategies detailed or a narrative description of the phenomenon created, respectively (Cairns and Cox 2008). The coding results should provide a direct basis for the next UCD phase, Conceptual Development.

Should limits be placed on the kind of information people can contribute to the wikimap? If yes, what kind of limits? How should those limits be enforced?

Paraphrased answer	Frequency (number of participants)
Yes	5
Keep contributions respectful/no obscenities	3
No	2
Keep limits minimal; prevent offensive or inappropriate posts only	2
Warn people that posts are visible to all, and let them self-police	1
Screen and validate information that comes in before posting	1
Based on focus of map	1
Nothing political/issue-based	1
No sacred sites	1
No sensitive ecosystems	1
Encourage contributions based on categories of information	1
Keep everything watershed-related	1
Only if children are encouraged to contribute	1

Figure 3: Open coding summary of participant responses to a Needs Assessment interview question for the case study project described in Sack (2013).

Stage 2: Concept Design

The second stage of the UCD process synthesizes the results of the Needs Assessment into guidance for the application developer. Although this stage has been termed ‘Conceptual Development’ by some Cartographers (Robinson et al. 2005, Bhowmick et al. 2008), that phrase more commonly is applied to the process of language acquisition during childhood, while the term ‘Concept Design’ is used in the fields of design, art, and architecture to describe an initial sketch or specification of a project idea.

The Concept Design should describe the layout, tools, information, and architecture of the application, including multiple mock-ups, or graphical concept prototypes, showing different views of the intended application interface. A mock-up example from Sack (2013) is shown in Figure 4. The Concept Design provides a sketch of what the application is intended to look like. Stakeholder feedback should be sought via regular meetings, phone calls, and e-mail communication (Robinson et al. 2005).

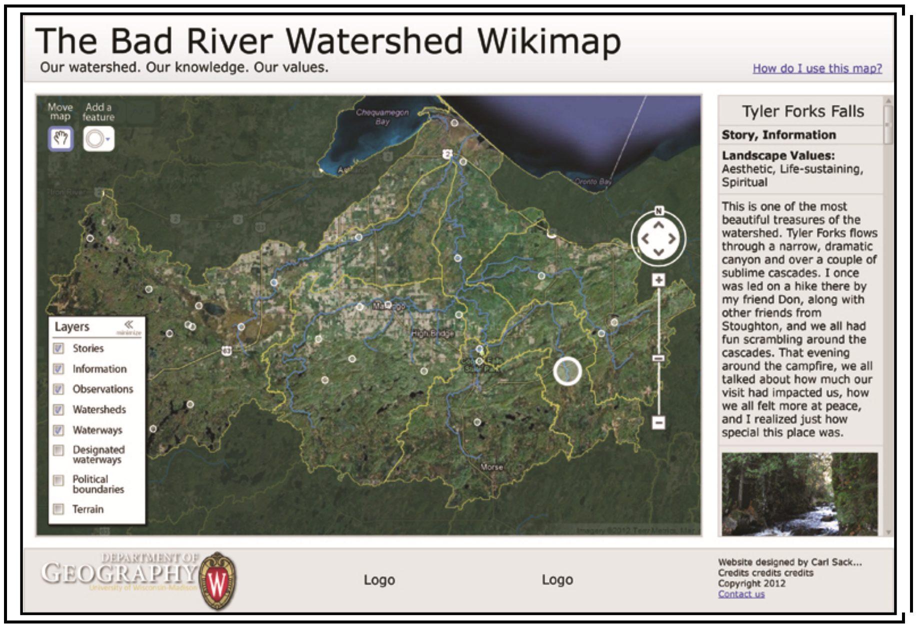
The organization of the Concept Design may follow that of the Needs Assessment questions, using either the five-level Garrett (2002) framework or a thematic approach. ISO 13407 specifies the following components:

- *Relevant users:* the range of those who will both contribute to and use the application;
- *Design goals:* a clear statement of what the application will do;

- *Priorities:* the order of importance in which different requirements will be met;
- *Benchmarks:* metrics against which the emerging design can be tested;
- *Acceptance:* evidence that the requirements have been accepted by stakeholders;
- *Statutory requirements:* any official standards or legal requirements that must be met;
- *Clear documentation:* a record of the requirements, including how they have changed over time.

Another approach to specifying requirements places them into three or four categories: user requirements, usability requirements, organizational requirements, and technical requirements (Maguire 2001, Medyckyj-Scott and Hearnshaw 1993).

- *User requirements* are descriptions of tasks that the system will support and the functions that will be provided to support them, along with example task scenarios, interactions, and features designed to support the context of use. These requirements may also include semantic structures, syntactic structures, support structures, users' hardware, and design constraints.
- *Usability requirements* are measurable objectives for the effectiveness, efficiency, and satisfaction of users.
- *Organizational requirements* are the tasks and conditions required to meet the social context of the application, derived from an understanding of power structures, motivations, and values held in the user community.
- *Technical requirements* are the system components that will be used to meet the requirements specified in the other three categories.


The Bad River Watershed Wikimap

Our watershed. Our knowledge. Our values.

Tyler Forks Falls

Story, Information

Landscape Values: Aesthetic, Life-sustaining, Spiritual

This is one of the most beautiful treasures of the watershed. Tyler Forks flows through a narrow, dramatic canyon and over a couple of sublime cascades. I once was led on a hike there by my friend Don, along with other friends from Stoughton, and we all had fun scrambling around the cascades. That evening around the campfire, we all talked about how much our visit had impacted us, how we all felt more at peace, and I realized just how special this place was.

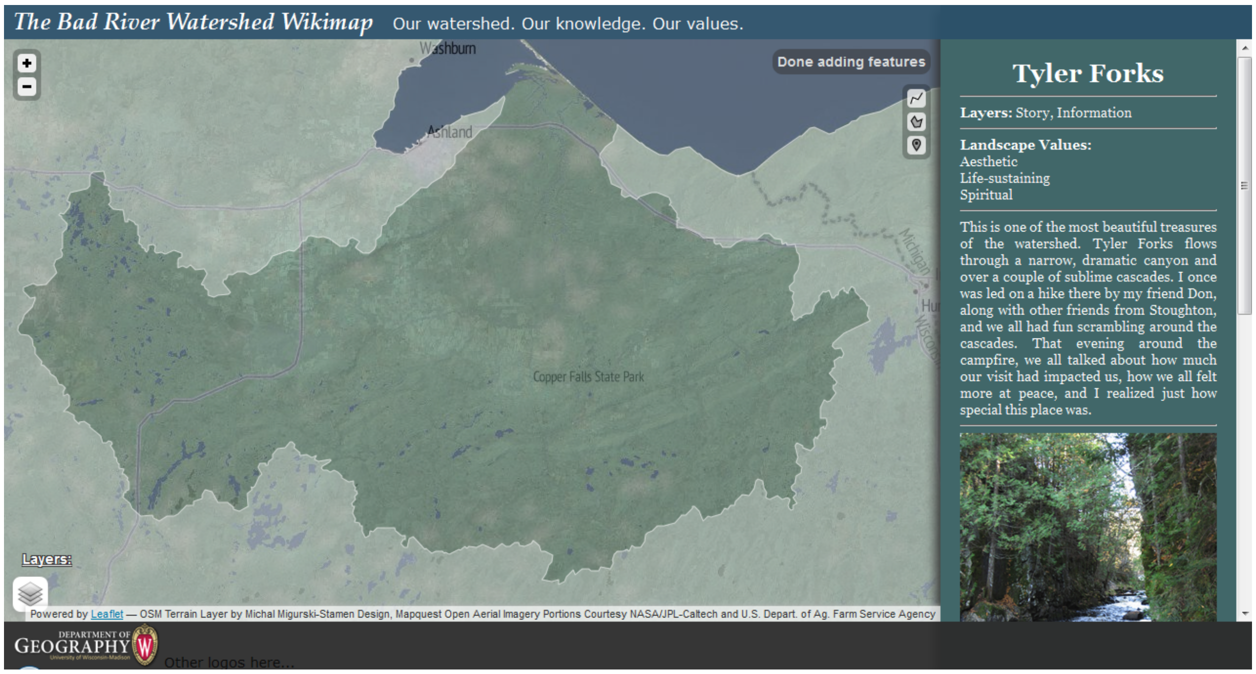
Website designed by Carl Sack...
Credits credits credits
Copyright 2012
Contact us

Figure 4: A Concept Design mockup produced for the case study project described in Sack (2013).

Once an initial Concept Design is completed, it should be reviewed by stakeholders and modified according to feedback received. The outcome should be specific enough that the intermediate and final versions of the application can be evaluated for consistency in implementing the specified design components. Designs may also be submitted to a usability expert for conformity assessment, determining if the planned system meets accepted usability standards or in-house guidelines of the developer's organization (Roth 2011b). Since the recruited group of stakeholders is unlikely to contain technical experts in programming and web development, it is appropriate for these users to be regularly consulted at this stage but not necessarily included on the core development team (Haklay 2010).

Stage 3: Prototyping

The third stage of the UCD process is the development of partially-working instances of the system that demonstrate how it will look and behave for gaining user feedback. A prototype can be a *throwaway* (an instance created only to demonstrate a specific feature, then discarded) or a *build-through* (a partial interface that can be used to produce the final system) (Medyckyj-Scott and Hearnshaw 1993). In the Sack (2013) case study, the development of both throw-away and build-through prototypes allowed for field trials of different technologies. It resulted in a change in the software stack used to construct the system, as well as tweaks to the interface design based on stakeholder feedback. One instance of the build-through prototype from the project is shown in Figure 5.



Tyler Forks

Layers: Story, Information

Landscape Values:
Aesthetic
Life-sustaining
Spiritual

This is one of the most beautiful treasures of the watershed. Tyler Forks flows through a narrow, dramatic canyon and over a couple of sublime cascades. I once was led on a hike there by my friend Don, along with other friends from Stoughton, and we all had fun scrambling around the cascades. That evening around the campfire, we all talked about how much our visit had impacted us, how we all felt more at peace, and I realized just how special this place was.

Powered by Leaflet — OSM Terrain Layer by Michal Migurski-Stamen Design, Mapquest Open Aerial Imagery Portions Courtesy NASA/JPL-Caltech and U.S. Depart. of Ag. Farm Service Agency

Figure 5: A build-through prototype of the case study wikimap application from Sack (2013).

Informal evaluation is one method of gaining stakeholder feedback the prototypes. While developing wikimap prototypes for the case study project, questions about the interface were sent to participants via e-mail. Feedback was not as robust as the formal Needs Assessment interview results, due to lower motivation among stakeholders and the informal method of the information gathering employed. A series of short phone interviews with stakeholders may have boosted participation.

A *stakeholder task analysis* could be used to gain even richer insight from potential system users. This procedure characterizes the actions and/or cognitive processes needed to achieve a task. Individual stakeholders are observed interacting with the prototype, and the sequences of actions and input/output flows are mapped as flow charts or other forms of notation (Maguire 2001).

Usability expert evaluation can also inform the evolution of prototypes. This procedure may entail greater financial cost, but is more rapid than a task analysis and give as much insight as is necessary at this stage. The usability expert assumes the role of the end user and completes a set of tasks with the system to test it against guidelines, heuristics, or their own previous experience (Sweeney 1993). *Guidelines* may include the features identified in the concept design, design consistency of the interface components, and/or a set of established design standards used for the project. *Heuristics* are well-accepted general usability principles, a subset of guidelines (Roth 2011b). The expert then makes qualitative recommendations for improvements to the system.

Stage 4: Formative Evaluation

Once a mostly-working system is available, a more formal set of usability studies should be run to gain insight into the system's likely real-world performance and make improvements accordingly. The Formative Evaluation process differs from summative evaluation in that its purpose is to identify and rectify system problems. By comparison, a summative evaluation quantitatively measures the performance of the final product against its intended functionality or the usability of similar systems (Maguire 2001).

Recruitment

Up to this stage, the primary actors involved in the process have been the developer(s) and stakeholders. A Formative Evaluation will most accurately reflect the application's real-world performance if the setting for it is realistic and the pool of participants is broadened to include a diversity of potential users. Recruitment is therefore an important task and should be carefully planned.

Community partnerships that were developed during the Needs Assessment can be especially useful for recruitment at this stage. The partner organizations' knowledge of the potential user community and ties to community leaders place them in a good position to reach out to the public. Community members may be motivated to participate if they feel their input will contribute to creating an interesting or helpful resource for themselves or their community. Community groups can promote the benefits of participation to their constituents.

Developers can also recruit participants through advertising and word-of-mouth. Effective recruitment techniques include the *snowball method*, in which participants are asked to recommend others to invite to participate, and setting up sessions in public venues where potential participants are already likely to meet and socialize (Cameron 2005).

Components

The content of Formative Evaluation sessions can be characterized in three ways: by the scope of the study, the procedures used, and the recording methods. Figure 6 diagrams how Formative Evaluation components relate to one another, with common component combinations represented by the thicker connecting lines. Combinations that use a participant observation procedure are connected with single-dash lines, while those that rely on a talk-aloud/think-aloud

procedure are connected with multiple-dash
lines.

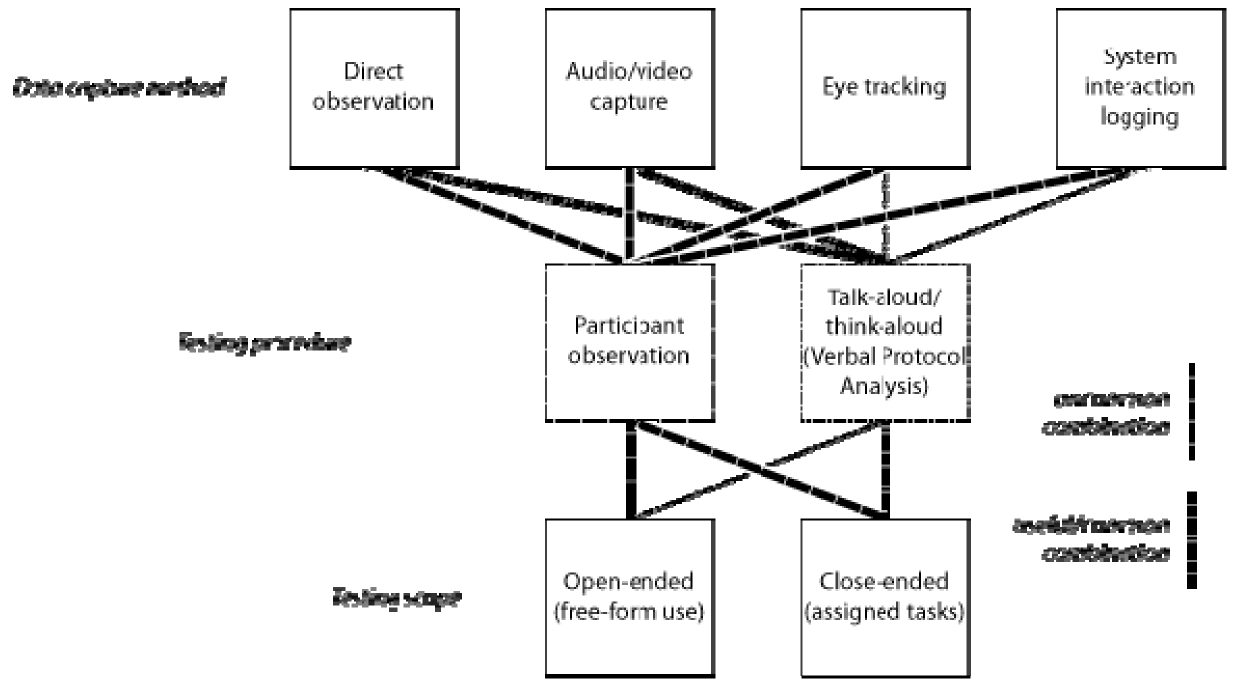

Data capture method
Direct observation
Audio/video capture
Eye tracking
System interaction logging
Testing procedure
Participant observation
Talk-aloud/ think-aloud (Verbal Protocol Analysis)
Testing scope
Open-ended (free-form use)
Close-ended (assigned tasks)

Figure 6: Formative Evaluation components diagram.

The *scope* of the evaluation can be open-ended or close-ended. *Open-ended* evaluation provides participants with initial training in the form of documentation or a demonstration, then allows the participants to use the system freely while being monitored by a facilitator or recorded. *Close-ended* evaluation involves participants being given a set of specific tasks that they are asked to complete. Typically, the accuracy and time to completion of each task will be recorded, along with observations of actions taken by the user.

While some structure may be used to test the effectiveness of the system for specific objectives, at this stage participants should be given some time to explore the system and ask questions, as their unguided experience may give insight into usability problems not anticipated by the developers. This occurred during the Formative Assessment in the Sack (2013) case study, when participants using the wikimap in an unstructured way found multiple bugs and suggested features that would add to the application's usability.

Typical *procedures* used for the Formative Evaluation include participant observation and talk aloud/think aloud. In *participant observation*, the facilitator(s) watch users interact with the interface, with the goal of generating a narrative of how the interface is used and any problems experienced by the user (Roth 2011b). The length of the observation can range from a brief, one-time session to long-term, ongoing study. In *talk aloud/think aloud* studies (also called *Verbal Protocol Analysis*), participants are asked to vocalize what they are doing or thinking and why while using the application. This method can help elucidate the cognitive processes of the user and determine how the user conceptualizes the application (Sweeney et al. 1993).

Every usability evaluation will require some method of *recording*. *Direct observation of the user* by the facilitator, who watches as the user interacts with the system and takes notes on any problems, is the simplest approach. This method has the advantage of requiring little in the way of equipment or setup and allowing for more open-ended dialogue between the users and developers, potentially creating as much insight as a more formal study (Sweeney et al. 1993). The tradeoff is a lower level of data detail and accuracy than can be achieved with a more formal recording process. Some participants may also find it unnerving being watched from behind.

Audio and/or video capture equipment can be used to augment or take the place of direct observation. The equipment need not be fancy; a simple camcorder set up on a tripod or a small audio recorder placed at the participant's work station are appropriate. Although many computer monitors now come with factory-installed cameras for videoconferencing, these are not ideal for recording usability testing, since they capture the user's face rather than what is happening on their computer screen. Recordings can be reviewed and coded for user interactions and performance metrics such as task completion time, accuracy or error rate, and learning rate (Medyckyj-Scott and Hearnshaw 1993).

Eye tracking can be used to gain detailed information about what interface features the users are attending to in real time. This method requires specialized recording equipment and software, necessitating greater cost and setup. Modern systems are non-invasive to the user. Eye tracking is most useful for close-ended evaluation, as participants that are given a task to engage with are more likely to be conscientious about what they are looking at, a phenomenon known as the eye-mind hypothesis (Cairns and Cox 2008). Data are interpreted by aggregating fixations, or places on the computer screen where the user's gaze came to rest for 200-300 milliseconds. The results can reveal design inefficiencies by showing what parts of the interface are underutilized and where users expect certain interface features to be located.

System-based interaction logging relies on functions built into the software under development, or the use of a separate piece of software, to record the individual operations performed by a user on the system. The results can reveal whether the system is primarily being used in the ways it was intended to be (Sweeney et al. 1993). This approach can be implemented for coastal web atlas tools through the use of PHP and AJAX, technologies that allow for two-way communication between a client browser and database without having to reload content on the web page. Functions can be added to the application script that record each interaction between the user and the interface and place operator codes or counts in a database for later analysis (Edsall 2003, MacEachren et al. 1998). This approach allows all users' interactions to be recorded, regardless of whether they use the application at home or in a testing lab. It was implemented in the wikimap application built for the Sack (2013) case study.

Variables

Depending on the type of evaluation completed, several variables can be coded for and analyzed in the results to determine necessary interface changes. These variables will be mentioned here; further analysis methods will be discussed under Stage 6.

Time to task completion can be used to determine whether the interface promoted efficiency in its use. This measure is especially helpful when comparing multiple versions of an interface to determine which is most efficient. Engineering models such as GOMS (Goals, Operators, Methods, Selection rules) can be used to hypothesize how long a given task should take with a given interface, which can then be tested against observations of users working with the system (Cairns and Cox 2008).

Error rate is a measure of interface effectiveness, or whether the interface allowed the user to complete the task correctly with minimal difficulty. This measure is most commonly employed using questions that require users to answer by completing a task with the interface. The error rate is the percentage of questions that were answered incorrectly. If the error rate is high across users, it may indicate flaws with the interface that are misleading or confusing to users (Medyckyj-Scott and Hearnshaw 1993).

User Satisfaction can be gleaned from survey questions asked as part of the Formative Evaluation or from open-ended user feedback. 'Satisfaction' is defined as the degree of positive emotion or attitude a user has toward an interface (Sweeney et al. 2003). Survey questions may probe user satisfaction through Likert scales ("rank the pleasantness of the experience on a scale of 1-5, 1 being least and 5 being most satisfying"), while open-ended feedback may be measured or guestimated based on the number of positive versus negative comments about the interface from users.

Learnability and *Memorability* are measurements of the user's abilities to understand the purpose of the application's features and to retain that understanding after a time lapse, respectively (Andrienko et al. 2002). Learnability can be estimated using error rates, survey questions, or open-ended user feedback. Memorability can be tested by asking the same users to complete the same application tasks after a period of absence from using the application.

Interaction operators are the actions provided by the interface that allow users to manipulate the display (Roth 2011a). They can be coded from video of the user interacting with the application or by system-based logging. Analysis of what interaction operators are used and when can reveal what application features are being used, how they are being used, and whether their use matches the original application objectives.

Bugs are unintended errors in the system that need to be remedied before the application can be considered fully functional. Formative Evaluation is extremely valuable for catching bugs that may have been missed by the developer. This is not a minor point; usability suffers and frustration rises rapidly when the user is prevented from completing a task by programming errors.

Stage 5: Debugging and Release

This stage entails applying the lessons learned during the Formative Assessment to the application. *Debugging* (removing programming errors to enhance application function, speed, and stability) is typically the most time-consuming task. Hopefully, the involvement of potential end users throughout the process has ensured that required changes to the interface will be relatively minor at this point. Debugging should really begin with the build-through prototypes and continue after the application's final release, dealing with issues as they arise. User feedback can be gained during this stage through issue tracking software, online forums, and follow-up e-mails and phone calls (Robinson et al. 2005).

Release, the point at which a stable version of the application is made publicly available, should occur only after the developers and design team are satisfied that all major bugs have been squashed and the application meets the majority of its objectives. However, it does not mean that the application is 'finished.' Useful software is 'alive' in the sense that it continues to be updated until it has outlived its purpose or something better has come along and it ceases to be maintained.

The application is more likely to succeed in gaining users if it is actively promoted during and after the Release phase. Adoption of an application usually begins slowly and is led by an enthusiastic subset of users. The transition point that separates successful from unsuccessful applications, called the 'chasm,' is the time period during which an 'early majority' of people with a potential interest in the application either will adopt or not adopt it (Moore 2006). Effective promotion can improve the adoption rate of the application by informing potential users that it exists and could be of use to them. During this stage of the Sack (2013) case study, for example, the wikimap application was promoted through public appearance; a laptop computer was set up to provide a demonstration at pre-scheduled public events attended by hundreds of people.

Stage 6: Summative Evaluation

The optional final stage of the UCD cycle is another round of usability testing. Unlike the Formative Assessment, the results of these tests are unlikely to be used to revise the application at hand. Rather, the purpose of the Summative Evaluation is to measure the extent to which the application meets user objectives and usability benchmarks in order to draw conclusions that may aid in the development of future applications (Roth 2011b). It need not occur in every implementation of the UCD process.

Data Analysis

Because the application has been released at this point, the usability testing that takes place at this stage should draw from the set of all end users, ideally at random. The Summative Evaluation may use any of the quantitative or qualitative methods previously discussed in Stage 1 and Stage 4. Rather than review each of these again, information is provided below about more in-depth data analysis that can be performed with each.

Interviews, Focus groups and *Delphi method* are the most formal and time-consuming qualitative methods of getting at the usability of the application. Because they are subjective, they are likely to be most useful for determining user satisfaction with the application (Roth 2011b). Open, axial, and selective coding (discussed in Stage 1) should be used to evaluate the session results. Results can be reported using a discount or anecdotal strategy, wherein quotes, paraphrases, and summary representing the leading opinions of participants are included.

Questionnaires are familiar to many people and can return rapid results. They can be used to collect qualitative and/or quantitative data. Likert scale questions can be statistically analyzed and graphed as histograms to visually represent the preferences of users. If more formal results are needed, a chi-squared test can be performed to compare the distribution of answers to a normal distribution and determine the probability of observing the difference between the results and what would occur randomly. If comparing two groups of users (e.g., novice users and experts), a statistical t-test can determine whether there is a significant difference between the performance or attitudes of the groups (Cairns and Cox 2008). The Sack (2013) case study utilized a questionnaire to gather data about user satisfaction and application use patterns. The latter type of data was compared to the findings of the interaction analysis to improve their reliability (see below).

Participant Observation and *Talk-aloud/think-aloud* studies can both be coded to reveal interaction operators, error rates, time to task completion, and user attitudes and opinions. Talk-aloud/think-aloud analysis can include coding for statements that indicate a cognitive process. Times and errors can be compared to hypotheses or to similar results from testing a former version of the application to determine whether there are significant differences or improvements. Attitudes and opinions can be reported anecdotally, or analyzed statistically if close-ended survey questions are included in the user testing. Interaction operators derived from the coding of video can be subjected to interaction analysis, described below.

The process of coding video recordings can be complex and time-consuming. A typical method is to apply a pre-determined coding scheme based on prior research or secondary sources, and have multiple personnel independently code each video according to the designated scheme (Roth 2011a). This process usually takes between 3 and 10 times the run time of the video per coder. Additional time is required for multiple sets of coding results to be evaluated for agreement between coders, an important measure of reliability of the coding. There are two types of reliability scores: percent fit, which reveals the percent of interactions codes that match between coders, and Cohen's kappa, which estimates percent fit after correcting for the proportion of chance agreement. Analyses with a score of 80% or higher are considered very reliable (Nyerges et al. 1998).

Interaction Analysis

Interaction analysis can be performed on interactions coded from video records or from system-based interaction logs, which are automatically generated and take significantly less time to process. Although there is relatively little literature on interaction analysis to date, it can reveal robust qualitative insight into how users are making use of an application and whether the application's use meets its original objectives.

Several statistical and visual methods have been employed used by researchers. Interaction codes can tabulated by *frequency* (how often the code occurs) and *extensiveness* (how many participants applied the code) to get an overall sense of which interactions were being performed most often under a given scenario. The results can be grouped by individual user to reveal differing interaction strategies between participants, or aggregated among groups of users or all users to show the relative frequency of interactions performed overall. These results are typically visualized in histogram form, sometimes color-coded to indicate categories of interaction and subdivisions with the categories (MacEachren et al. 1998, Andrienko et al. 2002, Robinson 2008, Roth 2011a).

Another analysis strategy is to display *interaction sequences* or *timelines* for individual users, revealing what pathways through the application were used. These sequence graphs can be compared against one another to determine whether particular interaction strategies led to success or frustration in completing a task (Edsall 2003).

The case study discussed in Sack (2013) analyzed system-based interaction logs both by frequency of recorded interactions and by interaction sequences. Interaction sequences were further divided into a *pairwise analysis*, which compared the relative frequencies of sets of two interactions that were performed in sequence, and *use session analysis*, which compared the sequences of the first 18 interactions performed on the application across all user interaction logs. These sequences were displayed using Sankey diagrams, which are visualizations designed to model product or energy flows between nodes of an industrial ecosystem. An example Sankey diagram for use session analysis is shown in Figure 7.

Three *use patterns* (sets of interactions that revealed a particular goal of application use) were delineated based on relative frequencies of different sets of interactions in each log. The session analysis revealed that some logs contained a single use pattern, and others contained multiple use patterns performed in a sequence (i.e., the user switched patterns mid-way through their use of the application). The overall analysis revealed that most wikimap users leaned heavily toward map reading and information seeking patterns, while few users took advantage of the ability to add information to the map—a primary objective of the application. This finding was confirmed by data collected from the user questionnaire discussed previously.

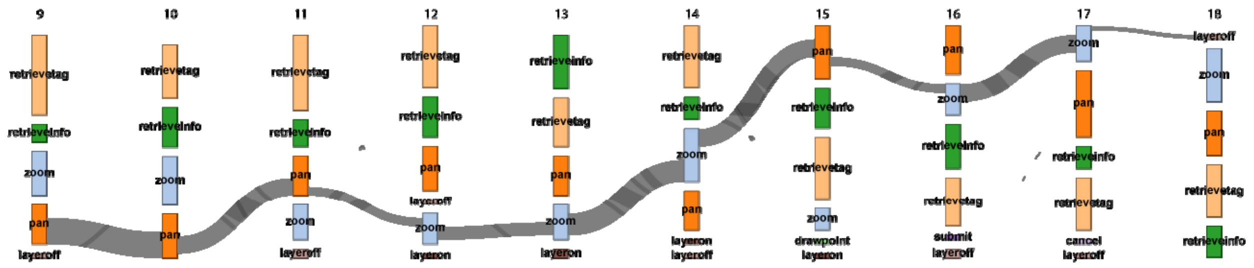


Figure 7: Sankey diagram from Sack (2013), showing interaction sequences from system-recorded logs, with one log highlighted that conforms to a 'map reading' use pattern. Numbers above the diagram represent the position of each set of interactions in the sequence, while each interaction code performed at that position is shown as a colored node, sized proportionally to the number of logs containing that interaction at that sequence position.

Conclusion

Good design makes a difference. For coastal spatial decision support tools, it can determine whether or not the tool is adopted by decision-makers and members of the public who are not computer experts. User-Centered Design puts those who are expected to benefit from the application at the center of the development process to ensure that the intended benefits are realized.

This paper has presented a number of social science techniques that are appropriate to implement at different stages of the UCD process. It is not meant to be comprehensive; to cover all possible variations of the process would take multiple volumes, and there are already several valuable resources, referenced here, that can be consulted for more detailed guidance. Nor should every procedure possible at each stage be implemented. Each development scenario differs based on the use context, intended outcomes, and time and resources available to developers. Thus, each should have an individualized development plan.

What makes UCD powerful is not the specific methods chosen at each stage, but rather the overall approach that emphasizes early and consistent feedback from users, the employment of empirical usability measurements, and iterative stages that allow for rapid modification of application design and prototypes. There is no one 'right way' to do User-Centered Design, but there are a number of wrong ways, which all involve not adequately consulting end users before the application is released. Methods may evolve as research continues to progress, but usability will continue to be a leading issue in web application development, and the principles of UCD should be implemented to avoid future design headaches.

References

Adrienko, N., Andrienko, G., Voss, H., Bernardo, F., Hipolito, J., Kretchmer, U., 2002. Testing the Usability of Interactive Maps in CommonGIS. Cartography and Geographic Information Science 29, 325–342.

Bhowmick, T., Griffin, A.L., MacEachren, A.M., Kluhsman, B.C., Lengerich, E.J., 2008. Informing geospatial toolset design: understanding the process of cancer data exploration and analysis. Health & Place 14.

Cameron, J., 2005. Focussing on the Focus Group, in: Qualitative Research Methods in Human Geography. Oxford University Press, Melbourne.

DiCicco-Bloom, B., Crabtree, B.F., 2006. The qualitative research interview. Medical Education 40, 314–321.

Edsall, R.M., 2003. Design and Usability of an Enhanced Geographic Information System for Exploration of Multivariate Health Statistics. The Professional Geographer 55, 146–160.

Gabbard, J.L., Hix, D., Swan II, J.E., 1999. User-Centered Design and Evaluation of Virtual Environments. IEEE Computer Graphics and Applications 19, 51–59.

Garrett, J.J., 2003. The elements of user experience: user-centered design for the web. American Institute of Graphic Arts ; New Riders, New York; Indianapolis, Ind.

Gould, J.D., Lewis, C., 1985. Designing for Usability: Key Principles and What Designers Think. Communications of the ACM 28, 300–311.

Haklay, M., 2010. Interacting with geospatial technologies. John Wiley, Chichester, West Sussex, UK; Hoboken, NJ.

Human Centered Design for Interactive Systems (ISO Standard No. 13407), 1999. . International Standards Organization (ISO), Geneva, Switzerland.

Human factors in geographical information systems, 1993. . Belhaven Press ; Distributed in North America by CRC Press, London : Boca Raton, FL.

MacEachren, A.M., Boscoe, F.P., Haug, D., Pickle, L.W., 1998. Geographic Visualization: Designing Manipulable Maps for Exploring Temporally Varying Georeferenced Statistics. IEEE Symposium on Information Vizualization 87–94.

Maguire, M., 2001. Methods to support human-centred design. International Journal of Human-Computer Studies 55, 587–634.

Moore, G.A., 2005. Crossing the chasm: marketing and selling technology products to mainstream customers. HarperCollins Publ., New York.

Morgan, D.L., Scannell, A.U., 1998. Planning Focus Groups, Focus Group Kit. SAGE Publications, Thousand Oaks, CA.

Nyerges, T., Moore, T.J., Montejano, R., Compton, M., 1998. Developing and Using Interaction Coding Systems for Studying Groupware Use. Human-Computer Interaction 13, 127–165.

O'Dea, L., Cummins, V., Wright, D., Dwyer, N., Ameztoy, I., 2007. Report on Coastal Mapping and Informatics Trans-Atlantic Workshop 1: Potentials and Limitations of Coastal Web Atlases. University College Cork, Ireland.

Research methods for human-computer interaction, 2008. . Cambridge University Press, Cambridge, UK ; New York.

Robinson, A.C., 2008. Collaborative Synthesis of Visual Analytic Results. GeoVISTA Center, Department of Geography, The Pennsylvania State University.

Robinson, A.C., Chen, J., Lengerich, E.J., Meyer, H.G., MacEachren, A.M., 2005. Combining usability techniques to design geovisualization tools for epidemiology. Cartography and Geographic Information Science 32, 243–255.

Roth, R.E., 2011a. Interacting With Maps: The Science and Practice of Cartographic Interaction (Dissertation). Pennsylvania State University, University Park, PA.

Roth, R.E., 2011b. A Comparison of Methods for Evaluating Cartographic Interfaces. Proceedings of the 25th International Cartographic Conference.

Roth, R.E., Harrower, M., 2008. Addressing Map Interface Usability: Learning from the Lakeshore Nature Preserve Interactive Map. Cartographic Perspectives 60, 46–66.

Sack, C.M., 2013. Mapmaking for Change: Online Participatory Mapping Tools for Revealing Landscape Values in the Bad River Watershed (Master's Thesis). University of Wisconsin-Madison, Madison, WI.

Slocum, T.A., Cliburn, D.C., Feddema, J.J., Miller, J.R., 2003. Evaluating the Usability of a Tool for Visualizing the Uncertainty of the Future Global Water Balance. Cartography and Geographic Information Science 30, 299–317.

Stakeholder engagement strategies for Participatory Mapping, 2009. , Social Science Tools for Coastal Programs. NOAA Coastal Services Center, Charleston, SC.

Sweeney, M., Maguire, M., Shackel, B., 1993. Evaluating User-Computer Interaction: A Framework. International Journal of Main-Machine Studies 38, 689–711.

Tsou, M.-H., Curran, J.M., 2008. User-Centered Design Approaches for Web Mapping Applications: A Case Study with USGS Hydrological Data in the United States, in: Peterson, M.P. (Ed.), International Perspectives on Maps and the Internet, Lecture Notes in Geoinformation and Cartography. Springer, New York, pp. 301–321.