Agile User-Centered Web Engineering:

The Holy Grail of Web Development

Jean L. Foster

Massachusetts Institute of Technology
ABSTRACT

New, rapid software development styles, also known as agile design processes, reflect the need of organizations to respond to the evolution of the marketplace while keeping their costs low. User-centered design (UCD) methods insure that products satisfy the needs of the customer and are easy to use. In this paper, the primary components of agile design processes and UCD processes as applied to Web development projects are described and analyzed. The strongest features and results of each process are identified and then merged together into a proposed new hybrid process referred to as Agile User-Centered Web Design. This hybrid process uses contextual enquiry during the requirements phase, prototype testing during the early design phase, and talking-protocol usability test iterations throughout. The paper concludes with an explanation of why this process would be a good choice for any organization creating web-based applications.
AGILE USER-CENTERED WEB ENGINEERING: THE HOLY GRAIL OF WEB DEVELOPMENT

For those of us in the business of web application development, whether as programmers, designers, usability specialists or managers, the possibility of a rapid development process that supports the creation of easy-to-use products is enticing. This paper will attempt to map out such a process by merging two modern processes that have cropped up over the past decade as a result of the growth of web services. The dot-com boom of the nineties not only inspired technological advances but introduced new processes and methods to create these technologies. Two such innovations are agile design and user-centered design (UCD).

Agile design encompasses a number of different design methodologies such as such as Extreme Programming (XP), SCRUM (Beedle et. al., 1998) and Agile Web Engineering (AWE) (McDonald & Welland, 2002). These are all variations on the primary themes of the Agile Alliance Manifesto (Agile Alliance, 2001) and all of them streamline traditional application development methods by reducing their long development cycles down to one month or even one week. According to David J. Anderson (2001) in his article in the uidesign.net webzine, “The era of the 2 to 5 year software project is almost dead. Increasingly we are asked to deliver more and more complex software for use over the Internet, in shorter and shorter product cycles.” Agile design enables software developers to quickly respond to changing customer demands or market changes, an important requirement for the volatile web industry.
The UCD field, on the other hand, promotes development methods that improve the usability of software and products, and has also become increasingly important during the e-commerce boom. Companies competing to sell their products on the web found that one way to gain the advantage in a highly competitive marketplace was to improve the customer experience. UCD proved to be a highly effective way of doing that. The e-commerce web site that was easy to use and inspired enough confidence for customers to entrust their credit card information in online transactions won the customers, their loyalty, and their money, and therefore was more likely to survive.

Proponents of agile design and UCD are currently operating with little overlap in techniques and methods. Integrating the two processes would create a new hybrid process that is stronger and more effective than either process alone, and could address some of the shortcomings of each. This paper reviews UCD and general agile process theory, with a closer look at the AWE process since it was specifically designed for web development. The strengths and weakness of each process are examined, commonalities are explored, and aspects of each that should be kept in the new hybrid process are identified.

STRENGTHS AND WEAKNESSES OF AGILE PROCESSES

The Manifesto for Agile Software Development (Beck, 2001) consists of twelve principles, several of which seem particularly well suited for web development. These are:

• Priority placed on customer satisfaction;
• Welcoming changing requirements, even late in development;
• Delivery of working software frequently;

• Use of multidisciplinary, self-organizing and self-governing teams whose members are in almost constant contact;

• Extensive testing.

According to McDonald and Welland (2001, p. 4), web development requires multidisciplinary teams working in short-development project life cycles of 3 months or less, which fits in with agile processes principles. Agile design’s short development cycles and quick and frequent delivery of working code an advantage for web development because new code or features can be delivered to end users almost immediately on the web as compared to the release of desktop software packages that require that physical media be created and packaged and delivered to the customer. Agile process cycles consist of bursts of coding called “sprints” that can be as short as a week or a month. This style makes it easy to change direction, if necessary.

There are other interesting features of agile processes worth mentioning. The agile development team essentially manages itself and all of the work. The creativity of the developers is highly revered and supported. Management’s sole role is to protect the developers from distractions and clear roadblocks from their path. In agile processes people are more important than processes, “face time”, and sometimes even program in pairs. Agile processes also depend on frequent testing throughout the development cycle, and customer evaluations to ensure that the project direction is correct. The AWE process in particular uses some “best practices” from traditional programming methodologies, such as code reviews,
unit testing, and small integrations, but does them continuously in the short “sprint” cycles instead of just once as in the larger traditional cycle (Nuckols, 2003).

There are several drawbacks to agile processes. One weakness is that they do not scale well to large teams and seem to work best on projects with 12 to 15 developers (Constantine, 2002). Perhaps the most serious weakness of all agile processes is that they give scant, if any, attention to the design of the UI (user interface) of the products. According to UI designer Larry Constantine (2002), “[...agile] methods are light on the user side of software. They seem at their best in applications that are not GUI-intensive.” This is a serious shortcoming of both agile and traditional development processes, and one that user centered design attempts to correct. Unfortunately, agile processes see user centered design methods such as UI prototyping and usability testing as being too time consuming to include in their short cycles.

STRENGTHS AND WEAKNESSES OF UCD

Over the past 10 years research in UI design has shown that bringing the user into the design process is the single most important factor in creating highly usable, efficient, and satisfying applications. Methods such as contextual inquiry and usability testing are two effective techniques for achieving these results. Contextual inquiry is a method used for gathering requirements that brings development team members into users’ work environments in order to fully understand their work tasks and goals. Analysis of this information then helps guide the design of the products and ensures the direction that the design is taking is correct. Talking-protocol usability testing brings ends users into a test lab to
perform tasks on prototypes or early working versions of the product while observers take note of the test participants actions and words. Such testing is most effective when members of the application development team participate as observers of the tests. Many developers are shocked to see how users struggle to use the applications they’ve created and come away with a new perspective on their application. Data gathered from observing the prototype tests is then analyzed and suggestions are made for how to fix any problems that were observed. Usability testing can continue to be iterated until the developers feel confident that the design is right.

These interactions with real users are the best way of ensuring that the product will contain the functionality users need and that the user interface will be easy to use. Without this sort of contact with end users, developers end up guessing what the user needs, and usually they guess wrong.

The major shortcoming of UCD methods and the primary complaint of agile process proponents are that the iterations of usability testing and revising are too time consuming to fit agile processes (McDonald and Welland, 2001, p. 43). In response to the need for speed, “discount” usability methods were developed during the height of the e-commerce era and some streamlined UCD techniques were created. Traditional usability tests required each round of testing to involve at least 8 test participants. Identifying, recruiting and scheduling the test participants are arduous, time consuming, and costly. Each test requires a test participant, a usability expert, usually a technician to set up the lab, a facilitator and observers
and lasts between one and two hours. Efforts to reduce these costs resulted in methods that relied on fewer test participants. Some research suggested that as few as 5 talking-protocol test participants will find 80% of the UI problems (Landauer & Nielson, 1993). Other discount methods include pluralistic reviews and heuristic reviews. In a pluralistic review one test is done with one real user test participant and members of the design team take on the persona of a real user while walking through the test tasks. Heuristic or expert reviews can be conducted in one session with at least 5 usability experts reviewing the UI and comparing it against guidelines or heuristics.

THE BEST OF BOTH WORLDS

What would the ideal software development process be? A process that is rapid, agile, and focused on delivering a product that is not only functional but is also highly usable and delivers an enjoyable user experience. How would this process look? A good place to start building the hybrid process would be with the commonalities of agile processes and UCD. Both processes promote frequent testing, and maintaining the flexibility to change direction in midstream if necessary (which is the goal of UCD iterations). Both UCD and AWE are built on the concept that there is value in customer involvement, although this means slightly different things to the two. AWE recommends having the customer involved in an evaluation phase. The term “customer” in this sense does not usually refer to an end user however, but to a customer representative and the evaluation may be evaluation of the listing of the set of
features and functions. UCD requires that real end users be involved in testing by trying to perform real tasks using the application. The distinction is fine but extremely important.

To continue to build the hybrid process take the rapid pace, support for creativity, and strong management support for the design team from agile processes. From the UCD process bring in the end user presence throughout the cycle from contextual enquiry, through prototype testing, expert reviews, and usability testing.

Figure 1 illustrates the AWE process life cycle beginning with either a new application or an existing application. The results of the test phase can begin another iteration if the tests have
found problems that need to be resolved and recoding is needed. Unlike most other agile processes AWE practitioners believe that it is essential to involve end users in the process; however, they propose that the only place this can fit in the process is during the evaluate phase. They also recommend that this not be done during every iteration because of the time and expense (McDonald & Welland, 2001). Unfortunately, this is too late in the process for UCD methods to be of value. This is a common mistake that is even made when UCD is worked into traditional software development processes. Sometimes usability testing is confused with quality assurance testing which is done to find software bugs before release. Usability tests uncover higher level problems, not bugs. The problem is that if the usability testing comes late in the process, most of the code is already written and so changes made in response to problems found in the testing are more difficult, time consuming and costly to make. Waiting until the end of the process also means that it is unlikely that there will even be time to make corrections before the product release.

Figure 2 shows how UCD methods could be worked into the hybrid process. Contextual enquiry is done as the primary component of the requirements phase, to ensure that the requirements are in line with real user needs. After the requirements are written, early, rough prototypes of the UI are designed before any coding is done. Usability testing with these prototypes confirms or corrects the design assumptions about how users will interact with the product. At this point coding begins. An heuristic review is done of the early
implementation. Again, any problems found are corrected and then the code moves on to the QA test phase, after which there is another round of talking-protocol usability tests. Any serious problems found at this stage would require another iteration of the process.

This hybrid Agile User-Centered process design works UCD methods into a simple, lean process that maintains the benefits of both UCD and agile processes. The use of discount
usability methods keeps the life cycle short and so should satisfy proponents of agile processes. The early introduction of user evaluation of the design ensures that any serious mistakes in the direction of the design are caught early enough to be fixed without having to rewrite code. This is an extremely important cost savings benefit since fixing design errors after code is written is always very costly. One could argue that the time spent on the UCD phases is time saved later on in costly rewriting of code to fix design errors.

How much time is added to the agile sprint with the introduction of UCD? If the above process were taking place in a one month sprint, the contextual enquiry might not add any substantial time since it is used in place of other requirement gathering methods. Using discount usability methods with small numbers of test participants, the prototype testing might take 2 days; one for the tests themselves and one for test materials creation and reporting following analysis of the tests. The heuristic review could be done in half a day, and the final round of usability testing might also take a day. If there was a usability professional on the project, even if only dedicated at 20% time, most of the preparations could be done concurrently with other project tasks so there wouldn’t need to be extra load placed on other team members or extensive disruption to the cycle schedule.

One of the benefits of introducing UCD into projects is that the code produced is more likely to satisfy customer requirements so less effort will be needed during the post-release or maintenance phase. Studies have shown that 80% of software lifecycle costs occur after the product release, in the maintenance phase. Of that work, 80% is due to unmet or
unseen user requirements. Only 20% of this is due to bugs or reliability problems (Karet, 1993). This is partly due to the fact that ripping apart existing code to fix problems is more difficult and time consuming.

AGILE USER-CENTERED WEB ENGINEERING: THE END OF THE QUEST

The entire software industry has been slow to realize the importance of the UI in the success of products. Both traditional software development processes and agile processes have put the code and architecture first, and consideration about how humans will interact with their code last. This has resulted in immeasurable frustration and productivity loss for users of these products.

UCD turns this model right side up and proposes that thinking about how the end users need to interact with the product should come first, and coding is then done to the requirements that result directly from project interaction with end users. This invariably produces products that are easier to use and have the correct functionality. This should then translate into more demand for the product, lower support costs for the released product, lower maintenance costs for the product, and would enable the development teams to focus on adding new enhancements and functionality for future releases. This is clearly a win-win proposition for developers and end users.

As end users become more sophisticated they will be less tolerant of unusable applications. The youth of today, for example, are entering the marketplace with higher expectations for technologies than their elders. They have grown up playing remarkably
sophisticated and responsive video and computer games and are likely to find more
traditional application interfaces clunky. As this generation moves out into the work world
they will become a major market for productivity and office products and the industry will be
forced to respond to their demands.

The good news for all users is that there are signs that the computer industry is
beginning to understand the importance of how their users need to interact with their
products. In a recent National Public Radio news broadcast about the slump in computers
sales over the last few years, HP’s Carla Fiorina acknowledged that the technology the
industry creates is still too difficult to use: "I think the technology industry has to acknowledge
that, whereas engineers and geeks might like complexity, that most business people and

There are other indications of a wellspring of interest in integrating UCD into agile
development processes. An upcoming Usability Professionals Association conference has
added a workshop called "UCD in the Age of ‘Web Years’, XP, and Agile Programming
Methods: Towards Agile User-Centered Design." This workshop will bring together
experience usability engineers and managers to share their experiences in UCD with "agile"
and shortened development practices. These seem sure signs that the industry will be seeing
more efforts like this in the coming years.
Are these initiatives evidence of a trend for incorporating UCD concepts into modern development processes? One can only hope so, since it would make the online world a better place for us all.
References


