

Usability and User Interface Design in XP



Table of Contents

I: Introduction	page 3
II: The Current Contenders	page 4
III: Importance of Usability and User Interface	page 5
IV: The Customer	page 6
V. What is a Designer?	page 7
VI. Holistic Agility	page 8
VII. Why Planning Games Go Bad	page 9
VIII. What's in it for the Developer?	page 10
IX. Closing the Loop: Usability Testing	page 11
X. The Process	page 12
XI. Conclusion	page 13
XII. References	page 14

I. Introduction

Agile processes like Extreme Programming (XP), widely touted for their developmental efficiency and code quality, have received fewer accolades in the realms of user interface elegance, cohesion and usability. This is quite simply because they address neither the importance of these attributes nor methods for attaining them. Additionally, the typical flag bearers of these causes (creative professionals such as user interface designers and usability engineers) have no roles defined within XP.

In common practice, usability testing often consists of a post-delivery analysis where issues are identified and analyzed. Of course, any significant action based on this information is likely too costly to undertake by that point. The creation of usable software must be done within the framework of the agile process. By building upon the XP tenets of communication, simplicity, testing and courage, good usability practices can support significant gains not only in end user acceptance and customer satisfaction, but in developmental efficiency and comfort as well.

II. The Current Contenders

Several recent methodologies have attempted to marry high-quality user interface design and agile processes. One of the best known is Larry Constantine's Usage-Centered Design (U-CD), which proposes a model-driven approach. U-CD is billed informally as a lightweight process, and Constantine has illustrated approaches to implementing it within an XP framework. U-CD emphasizes the strength of User Roles, Essential Use Cases and Task Maps in providing a solid navigation architecture upon which functionality can be built iteratively.

Alan Cooper's Interaction Design has similar stated goals, and introduces the idea of a separate interaction design team that works as a trusted intermediary between customers and developers. However, as vividly illustrated in a debate with Kent Beck, Cooper's argument for a sizeable up-front design session doesn't sit well with XP's directive to jump right in and start coding, or with its fully iterative approach. Both of these methodologies offer valuable ideas of how User Stories could be better conceived, written and delivered. However, neither addresses certain particulars of realizing their objectives within a true agile framework. The most significant of these is the nature of the team required, the interactions between roles, and the tactical execution from design to delivery.

III. Importance of Usability and User Interface Design

To a user, the interface is the application. Naturally, the functionality behind it must be relevant and work properly, which are key benefits of XP. However, if the application is difficult to use or aesthetically unpleasant, satisfaction and acceptance will suffer in direct proportion to the level of frustration. Confused customers turn to technical support services, incurring additional costs for the provider, assuming they have the patience to complete their tasks at all. Poor usability costs money.

As the most externally visible part of a system, the user interface tends to evoke strong opinions. Once a screen has been designed to consensus, the process of changing it is often politically charged and painful, however worthy the cause. More importantly, users learn the system through the interface, and even minor refinements can be disruptive once familiarity is established, a fact to which anyone who has used a major application through multiple versions can attest. As such, iterating the user interface is a delicate art, requiring an approach different from that of programmatic refactoring.

XP emphasizes code quality; it is of paramount importance to leave your customer with a system that works as intended. Leaving a useful and usable system behind is equally vital. Useful, for the simple reason that if users don't accept the application, then clients will not realize the promised return on investment, and all of the development team's quality work will be wasted. Many studies have been done on usability's impact on the bottom line, but they all underscore the same fact: if something is too difficult to use, it will fail.

IV. The Customer

Many projects assume that “the customer” consists solely of those stakeholders who sign the checks. While these people are certainly important to please, and rightly define project scope and feature sets, they are often least representative of the system’s end users, and hence not necessarily qualified to design a system that meets those users’ needs. Product and project managers generally have the best of intentions and a strong knowledge of the subject domain, but the preconceived notions that accompany such familiarity can lead to a system mirroring more the establishment than the goal. Perhaps most importantly, internal agendas have a way of coloring an application’s goals in rainbow hues. However well an application fulfills its vision, it will fail if that vision does not suit its intended audience.

Recommendation:

“The Customer” should be defined to include not only project stakeholders, but end users as well. A steady user feedback loop, from story creation through user acceptance testing, can serve to dispel Customer uncertainty around feature sets, implementation and design. Their input on feature relevance and usability will validate design decisions and strengthen perceptions of iterative progress, leading to a better application and a happier Customer.

V. What is a Designer?

At a high level, there are two types of “Designers;” those originating in one of the various aesthetic fields of graphic design, and those hailing from the academic camps of human-computer interaction, usability engineering, ergonomics and human factors. The successful Designer in an agile team must draw from skills fostered in all of these fields and maintain a solid grasp of both the technical and business domains within which he or she will be working.

In the real world, Designers often embody many of the same traits as the worst customers: they throw requirements over a wall without the slightest inkling of the impact their requests will create. Furthermore, they rarely invest the effort necessary to address these lapses in comprehension. Designers (and Customers) often experience similar problems with developers, failing to understand why seemingly simple requests are met with incredulity and resistance.

In order to effectively bridge the gap between Customers’ ill-defined needs and the specific data needed to translate them into code, a party with knowledge of both sides is required. This means someone that understands the basics of programming, business and application domains, interaction design principles, usability principles, and aesthetic design principles, and is able to apply this knowledge tactically, possibly defending it from a skeptical audience.

Clearly, this is not a common set of attributes; however, the necessary skills can be acquired by the simple mechanism of open communication. The customer knows their business, and describing it in detail can often help them to clarify their present directives. The developer knows what requests are problematic and why, and is generally more than willing to share this knowledge. The keys are a willingness to listen and an ability to adapt, necessary qualities in any successful member of an agile team.

VI. Holistic Agility

XP tends to promote a very tight focus; don't worry about what's coming, just code the card you're holding. From a developmental perspective, this works, thanks to the magic of continuous integration, unit testing and refactoring. Unfortunately, once an application reaches a certain level of complexity, this approach can lead to disjointed, awkward or unnecessarily complicated interfaces designed around back-end functionality rather than the user's end goals.

Designing an efficient and elegant user interface requires some conception of what steps comprise a given task, and how tasks interrelate to create an application's flow. The manner in which these relationships are executed has the greatest impact on how well an application will behave to the user. In the context of XP, there is a tendency to view such statements as harbingers of "big up-front design." In practice, it is simply a matter of designing a few steps ahead of development, in quanta just large enough to allow for composition of a coherent interface for any given functional task. The Designer typically creates a sizeable queue within just a few iterations by continuing to work with the Customer to develop new stories while current ones are being implemented, reducing the opportunity for lag time and allowing for additional refinement prior to story delivery.

Recommendation:

During Release Planning, Designers work with the Customer to identify, define and prioritize the User Roles that will interact with the system. Next, they create and prioritize task maps based on these roles. These two items will help to clarify the overall vision of the product, and are iterated along with everything else as the project proceeds. Also, in the proper form², they can be used as direct inputs into object-oriented development.

This approach allows for a coherent navigational architecture and cleanly constructed task flows. It also provides a point of strategic focus, minimizing the wish-list effect and helping center requirements around actual user needs.

VII. Why Planning Games Go Bad

Have you ever been in a Planning Game where it seemed that cards were morphing before your very eyes? The Customer brings in a nice, neat stack, then proceeds to waffle wildly over what each card means, or how these two cards relate to that one, while the development team sits across the table quietly wondering why all this couldn't have been sorted out beforehand, so that they could commence estimation.

Problem Number One:

Customers often need help to understand, verbalize, visualize and organize their requirements. This relates closely to the real-world underpinnings that must exist for XP's principle of courage to succeed – namely, talented experts to back it up. While Customers may have only the vaguest notion of how to implement their vision, they have to hand over cards with a certain level of detail, and this inevitably generates some fear, uncertainty and doubt on their part as to precisely what their requests will yield.

Problem Number Two:

The conversion of requirement to interface is implicitly assumed to take place within the estimation process, and by developers with little opportunity to consider how exactly this interface will work, and generally limited expertise in designing it at all. They must take generally fuzzy cards and do their best to deliver concrete estimates, with a healthy dose of uncertainty on both sides of the table.

Recommendation:

Before the Planning Game occurs, bring in someone that understands both sides – the Designer. First, Designers work with the Customer, talking through needs and translating them into interfaces interactively. Bi-directional feedback during design sessions helps to assure that the Customer's needs are met, and the more fully realized designs lead to clearer cards. Most Customers appreciate expert consultation, as they can focus more on exactly what they need rather than how it will be implemented, without feeling that the former will preclude their involvement in the latter. Then, when the Planning Game occurs, Customers have a very clear idea of what they want, and Developers have cards that need little interpretation and lend themselves to immediate coding.

VIII. What's in it for the Developer?

Developers are often not entirely comfortable with user interface design. Deciding where an object is located on a page, how it will appear and how it will behave from a user's point of view tends to be a secondary concern when there is real coding to be done. Designing an optimal interface for a complex piece of software demands dedicated effort and experience, and programmers' time is too valuable to be spun away as they attempt to figure out the vagaries of behavioral flow and page layout.

Recommendation:

After Designers have worked with Customers to define an iteration's stories, they pair with the developers to implement them. This is like having an on-site customer that actually contributes to coding. Project managers facilitate an overlap of skills and a constant exchange of viewpoints between all parties. The open channel of communication helps Designers understand the basic impacts of their design choices, and helps developers provide an optimal implementation and get quick answers to front-end questions. In the end, the loads of both camps are lightened by the natural utilization of their respective skills.

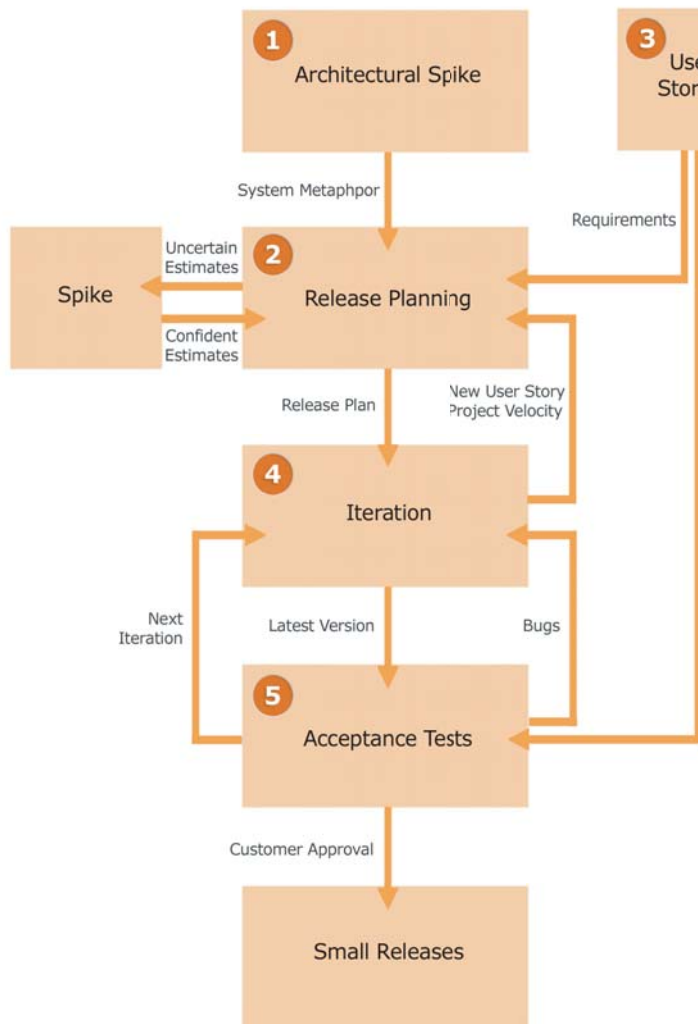
IX. Closing the Loop: Usability Testing

While including end users in the design process increases an application's chance for success, it is also true that people often don't know what works until they see it and use it. Agile processes are perfect fits for usability testing, as they allow teams to take immediate action based on the feedback received from users, maximizing the benefit of each iteration and reducing long-term costs.

Recommendation:

Include usability analyses within the cycle of User Acceptance Testing, and use the resulting feedback as fodder for the next iteration's designs. Usability testing is best performed with 4-10 end users in a moderated session, where designs are presented along with a list of tasks, and notes are taken to record and problems that occur when completing these tasks. This session can be followed by an open session where users voice particular concerns or requests. Once the data has been gathered, it is presented to the Customer along with recommendations as appropriate. The Customer then creates new cards and prioritizes them based on this information. This process helps to insure that users are getting what they want and need, and in the most usable form possible.

X. The Process



Engage end users & Define Initial User Roles and Tasks.

Determine whether end users will be on-site or on call. Designers might do contextual observation of users at this point to assist in defining user roles and determining how features and interfaces might mimic or differ from existing business processes and applications.

Solidify and prioritize User Roles and Task Cards.

Cluster similar cases and identify how they interrelate. Prioritize cards based on how typical, frequent, or important they are to the user or provider.

Translate requirements into user interface, then into User Stories.

Designers should pair with customers to interactively design screen mockups, which can then be used to write cards for more confident estimates.

Pair with developers to implement front end.

Designers should have a strong grasp of the specifics of front-end interaction, and be able to assist developers in defining and coding behaviors and page layouts.

Conduct usability tests with end users.

Small groups of users should be presented with the current application and a set of relevant tasks to complete. Any issues should be recorded and presented to the Customer for potential resolution in the next Iteration.

XI. Conclusion

Creating a more useful and usable application increases its chances of success upon release. A more successful application makes for more satisfied Customers, reflects well upon developers, and bolsters opportunities for repeat business. Rather than being an onerous task, accounting for usability can actually serve to boost both comfort levels and velocity throughout a project. The basic mechanism for this is an expansion of communication between Customers, Designers and developers to take advantage of the core competencies of each group.

XII. References

- [1] Beck, K. 2000. *Extreme Programming Explained*, Reading, MA: Addison-Wesley.
- [2] Robert Biddle, James Noble, Ewan Tempero. *Essential Use Cases and Responsibility in Object-Oriented Development*, Victoria University of Wellington.
- [3] Constantine, L. L. 1999. "Process Agility and Software Usability: Toward Lightweight Usage-Centered Design," Constantine & Lockwood, Ltd.
- [4] Constantine, L. L. and Lockwood, L. A. D. L. 1999. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*, Reading, MA: Addison-Wesley.
- [5] Cooper, Alan, 1995. *About Face: The Essentials of User Interface Design*, Boston, MA: IDG Books.
- [6] Nelson, Elden 2002. *Extreme Programming vs. Interaction Design*, Fawcette Technical Publications.

AUTHOR INFORMATION

Arlen Bankston
CC Pace
4100 Monument Corner Drive
Suite 400
Fairfax, VA 22030
703/631.6600
703/378-1589 FAX
arlen.bankston@ccpace.com
www.ccpace.com