Embracing Agile Development of Usable Software Systems

Jason Chong Lee
Center for Human-Computer Interaction and Department of Computer Science
Virginia Tech
Blacksburg, VA 24061-0106
chonglee@vt.edu

CHI 2006, April 22–27, 2006, Montréal, Québec, Canada.
ACM 1-59593-298-4/06/0004.

Abstract
The interdisciplinary nature of system design can lead to communication problems between developers in different fields. This is becoming evident in the emerging field of agile software development which has largely ignored or been unable to address usability. This work presents a development process and toolset that draws on extreme programming—an agile software development process, and scenario-based design—a usability engineering process. This approach will allow developers in both fields to better communicate and work together to efficiently design usable systems.

Keywords
extreme programming, scenario-based design, agile development, usability, central design record

ACM Classification Keywords
H5.2. Information Interfaces and Presentation: User Interfaces – *Evaluation/Methodology, Theory and Methods*; D2.2 Software Engineering: Design Tools and Techniques – *Evolutionary prototyping, User interfaces*

Introduction
The growing importance of computing systems in our daily lives is reflected by the reality of the multi-disciplinary nature of system design. One critical

example of the problems that can arise from this is the disconnect between software engineers and usability engineers. Software engineers focus on the design, implementation and maintenance of software systems, but often marginalize the design of the human-computer interfaces through which those systems are used. On the other hand, usability engineers focus on developing systems so end-users can use them effectively but do not account for the underlying system design, implementation or market-driven forces that guide much of software engineering.

This research will address the problems associated with multi-disciplinary system design—focusing specifically on software and usability engineering—by developing a design process that supports the effective creation of usable software-based systems through common practices and toolsets derived from both areas. This work draws from extreme programming (XP), an agile software development process, and claims-centric scenario-based design (SBD), a usability engineering process. Although the two processes emerged from different fields, they share similar motivations and the combination of the two will result in a process that supports team collaboration in effectively developing usable computer systems.

Need for Agile Usability

Many existing software engineering processes are unable to account for continuous requirements and system changes requested throughout the development process. Agile software engineering methodologies have emerged in the last decade to address this shortcoming. Extreme programming, one of the most widely practiced agile methodologies, eschews large upfront requirements and design processes in favor of an incremental, evolutionary process [1]. It is structured to efficiently handle emerging and changing requirements through active *communication* among project stakeholders, *simple* design, *continuous* feedback from clients and from testing, *courage* through a willingness to address any needed changes and *respect* for team members.

One shortcoming with current agile software development methods is that they do not address usability [2]. Although this is an acknowledged shortcoming, it is not clear how to incorporate usability into such processes without sacrificing the benefits provided by agile software development. Two primary issues this work will address are:

- How can interaction architecture issues be addressed in an agile development framework?
- How can usability evaluations be conducted in an agile framework without bottlenecking the development process?

The first question has been addressed by combining a design representation based process from usability engineering to the extreme programming framework. The proposed work will address the second question.

Scenario-Based Design to the Extreme

Interaction architecture level issues are addressed by incorporating my past work on design representations in scenario-based design into the extreme programming framework. Scenario-based design is an iterative, user-centered process—qualities that are shared by many agile methodologies including XP. Our adaptation of scenario-based design is supported by LINK-UP, an integrated design environment and

knowledge management system [5]. A core feature of LINK-UP is the central design record (CDR), a design representation that makes explicit where and how handoffs occur in the development process and highlights design decisions that need reconsideration during subsequent development iterations [4]. The CDR consists of an organized set of scenarios describing different usage situations, claims which highlight positive and negative effects of specific design features, and overall design goals. In several studies, the CDR was found to help usability designers to iteratively improve designs through targeted feedback and support communication of design intentions to project stakeholders [3], [4].

Pilot Case Study Results. The purpose of the pilot study was to evaluate how effectively the CDR allowed developers to address interaction architecture issues in an agile development environment. Seven undergraduate students worked in teams of two and three to develop applications on handheld devices. Students iteratively developed CDR representations of their designs within the existing extreme programming framework to address usability issues. Extreme programming includes accelerated design sessions known as 'planning games' to determine key features to support and to identify and assign development tasks [1]. The CDR was initially useful to the student developers in defining the task flow and specific interface features. Both new claims and existing claims from LINK-UP were used to highlight specific design tradeoffs. However in subsequent iterations, developers did not use the CDR and instead focused their efforts on the functioning system prototype. Although this is in-line with the XP concept of design representations as transient artifacts, much of the usability benefits of the CDR and LINK-UP were lost.

Proposed Work: Agile Usability Evaluation

In the pilot study, the utility of the CDR diminished because of the lack of a clear connection between interface features described and usability evaluations that were conducted on it. A key addition to the process will be to define how this connection will be made to support iterative enhancement to the interaction architecture based on usability evaluations. By extending the concept of test-driven development to user interface testing and integrating it with the CDR design representation, developers will be able to validate system usability and iteratively enhance designs without requiring extensive up-front requirements analysis or interaction architecture design. Rather, requirements analysis will proceed hand-in-hand with interface design as requirements emerge and evolve.

One of the key concepts of extreme programming is test driven development [1]. When implementing features, developers first write test cases to validate the functionality of new code classes and methods before beginning any actual implementation. This growing suite of tests is used to verify code functionality. Only code that passes the entire test suite can be added to production code. This allows developers to make code modifications based on changing requirements while ensuring that all other parts of the code still function correctly. This process gives developers continuous feedback and allows them to reliably make changes to code at all stages of development. A similar process could work with respect to usability evaluations while still supporting

overall architecture views and consistency through the CDR. The organization of the CDR naturally supports both high level system views (through core claims and scenarios) and feature-level views (through information and interaction claims). Developers can leverage this view to develop usability tests to validate specific claims in the CDR. As developers associate usability tests with the claims in the CDR, they will build up a test suite analogous to the one used to validate program code. Both analytic and empirical evaluations from the evolving test suite will validate the features defined by each claim or identify areas for improvement in the interface that have to be made.

One important difference between the usability test suite and the code test suite is that the usability test suite cannot be completely automated since many evaluation techniques require participant feedback and data. As the design progresses, the number of claims and associated usability tests that need to be run would become difficult to manage in an agile framework. One way to mitigate this problem will be to use the relationships between the claims to identify which areas of the interface need to be evaluated in the current iteration [6]. A second way to mitigate this problem will be to leverage discount usability methods to validate claims at each iteration. These methods would be easier to run in short time-frames while providing useful usability feedback. Empirical evaluations, which are often more time and resource intensive, could then be spaced out between several iterations as necessary.

Key Contributions

Integrating usability evaluations into the agile development process by leveraging the CDR will allow developers to validate the quality of the system interface in an efficient manner using a design representation-based development process. The proposed solution will make the following contributions:

- Allow developers who use agile software development processes to efficiently address usability issues
- Support collaboration between software engineers and usability specialists by facilitating communication of design intent and rationale
- Support efficient design representation-based development by leveraging techniques from agile software development

References

[1] Beck, K. *Extreme Programming Explained: Embrace Change.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1999.

[2] Constantine, L. Process Agility and Software Usability. *Software Development 9*, 5 (2001).

[3] Lee, J. C., Chewar, C. M., and McCrickard, D. S. Image is Everything: Advancing HCI Knowledge and Interface Design Using the System Image. *In Proc. ACMSE 2005,* Vol. 2, 376-381.

[4] Lee, J. C., Wahid, S., Chewar, C. M., Congleton, B., and McCrickard, D. S. Spiraling Toward Usability: An Integrated Design Environment and Management System. Technical Report TR-05-15, Computer Science, Virginia Tech, 2005. (Submitted to Intl. Jour. of ITSE)

[5] Rosson, M. B. and Carroll, J.M. *Usability Engineering: Scenario-Based Development of Human-Computer Interaction*. Morgan Kaufman, New York, NY, 2002.

[6] Wahid, S., Allgood, C. F., Chewar, C. M., and McCrickard, D. S. Entering the Heart of Design: Relationships for Tracing Claim Evolution. *In Proc. SEKE 2004*, 167-172.