

Finding a Place for Discount Usability Engineering in Agile Development: Throwing Down the Gauntlet

By David Kane, SRA International (david_kane@sra.com)

1 Abstract

Many software development organizations are reporting great success with agile software development techniques. However, few techniques explicitly incorporate usability engineering. Further, some agile techniques may not address certain kinds of usability problems, e.g. they may not address the needs of novices as well as expert users. While there are many techniques for usability engineering, discount usability engineering should be of particular interest to the agile development community because the two disciplines share many of the same underlying principles. The challenge for the agile development community is to find ways to incorporate and gain value from such discount usability engineering practices.

2 Introduction

Many software development organizations are reporting great success with agile software development techniques. However, none of the major agile development methods explicitly incorporate usability engineering practices. The article index on the Agile Alliance web site illustrates the situation. There is no category for articles addressing usability, no articles that list usability in their title, and only one article in which usability appears in the summary. [1] There is some work that attempts the bridge the two disciplines. Larry Constantine has published an article and white paper relating his Usage-Centered Design approach to agile development. [2][3] Another white paper examines the role of usability testing as part of acceptance testing in Extreme Programming. [4] Experience reports have mentioned improved usability as an outcome or motivation. [5] However, for the most part, there has been little examination of potential convergence, and this represents a missed opportunity for the agile development community. [6]

Usability is an important attribute for software. For corporate applications, improved usability can yield significant cost savings by improving the productivity of users. For software with good usability, less time is spent learning and

using the software, support staff field fewer questions, and fewer errors are introduced into operational systems. For e-commerce applications, usability can be the difference between customers spending money or leaving a site. [7]

The field of usability engineering encompasses a wide range of practices, many of which would be difficult to apply in an agile development environment. The practices of discount usability engineering reflect the principles and spirit of agile development. Jakob Nielsen coined the expression “discount usability engineering” to describe a collection of simple, low-cost techniques for designing and testing systems for improved usability. [8] Incorporating discount usability engineering techniques as part of an agile development methodology can improve the usability of software.

In response the Agile Development Conference’s call for papers to challenge the agile community to reexamine its assumptions, stretch its thinking, or develop new approaches over the following year, I am throwing down my gauntlet. [9] I challenge the agile development community to find ways to incorporate and gain value from discount usability engineering practices.

3 Discount Usability Engineering

Discount usability engineering is an approach popularized by Jakob Nielsen. [7][10][11][12][13] While designing systems to be usable and assessing system usability can be an expensive endeavor, Nielsen argues that significant value can be gained by applying simple, low-cost techniques for design and testing. While the term discount usability engineering is usually associated with Nielsen’s work, others in the field have also described methods similar in spirit. Constantine has described variations of his Usage-Centered Design methodology that are simpler, faster and easier to use. [14] The goal of these approaches is not to find the absolute best, most usable design, but to make the usability of a system good enough to yield valuable user productivity improvements or additional customer

transactions. In addition, the techniques do not require advanced usability engineering training and tools, but can be applied by software engineers with a minimal set of tools and training. Nielsen describes a number of such practices including scenarios, simplified thinking aloud, heuristic evaluation and card sorting. This is a representative, but not a comprehensive list of discount usability engineering practices.

3.1 Scenarios

Scenarios are simplified prototyping approaches for eliciting user feedback. It is a usability testing approach that distills the system to the minimal essential elements needed for useful feedback. The system does not need to be either functional or complete, but have just enough capability and description to simulate the user experience for some user-driven activity, i.e. a scenario. [11] Usability testing with scenarios does not require functioning software. Low fidelity prototypes using paper can yield useful usability data very inexpensively. [15]

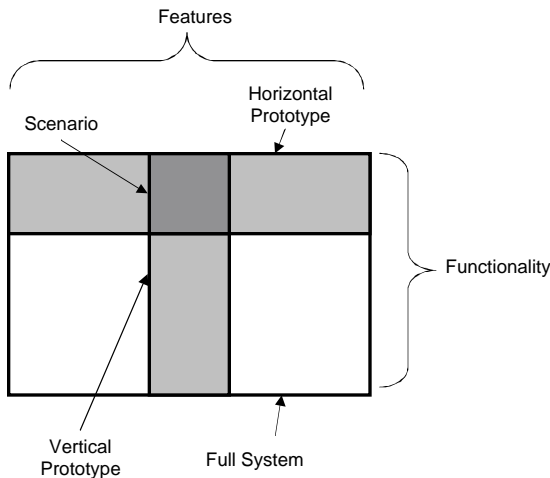


Figure 1: Scenarios Are at the Intersection of Broad and Deep Prototyping Approaches. [11]

3.2 Simplified Thinking Aloud

Simplified thinking aloud is an interview technique used to improve the feedback generated from usability testing. With this approach users are prompted to speak out loud their thoughts about what they are doing and expecting as they are evaluate a piece of software. Thinking aloud can be used when assessing with functional as well as paper prototypes. [11]

3.3 Heuristic Evaluation

In addition to techniques for usability testing, there are also approaches for improving usability

before testable elements are presented to users. Heuristic evaluation is an approach used by the developers to improve the usability of software by applying a small collection of usability principles to the design and development of the software. Some typical examples of heuristics include: [13]

Maintain Visibility of System Status: Users should always know what is going on in the system and receive feedback.

Enable Users to Rely on Recognition instead of Recall Memory: Available user options should be visible. Users should not need to memorize information from one dialogue to the next. Instructions should be visible or easily retrievable.

Emergency Exit: It should be easy for users to leave an unwanted state of the system without navigating extended dialogues. The system should support undo and redo capabilities.

These guidelines can be applied both as the engineers are developing the system and during periodic reviews of the software,

3.4 Card Sorting

Different users do not see an application or an application domain the same way. This can lead to usability defects in a software system if these different viewpoints lead to a mismatch between system assumptions and users assumptions. [20] Card sorting is one technique for finding users' mental models of the information space of an application domain. [12]

The gist of the technique is that each system feature or concept is written on an index card. Users are then asked to group the cards in piles that the user sees as similar and then to label each pile. The approach can be used to identify the major content categories of a web site, or may be used to organize system functions into a useful collection of menus.

4 Common Principles

There are a number of similarities between discount usability engineering and agile development in both spirit and tools. Nikula and Sajaniemi argue that the two are similar in that they both attempt to distill the minimal essential elements of their respective disciplines and focus on those keys. [16] The Agile Manifesto describes a set of values that underlie the various agile development techniques. [17] The ideas

and practices of discount usability engineering map well to these principles.

4.1 Individuals and interactions

Understanding users and their relationship to the software and environment are at the heart of most usability engineering techniques. Discount usability engineering seeks this understanding with practices that are simple and low-ceremony, and avoid complex tools and processes. Instead of testing in an expensive usability testing lab facility, these techniques can often be conducted in a user's or developer's usual work environment. The heuristic evaluation guidelines are similarly user-focused. [12] The heart of discount usability engineering is how to make individual user interactions with a software system more pleasant and effective.

4.2 Working software

Working software is very important for discount usability engineering, but its form and motivation differs from agile development in practice. In contrast, in agile software development, working software means that computers are able to verify the performance and capability of the software. For usability engineering working software means being able to understand the interface between the system and the users. Since users are much more flexible and adaptable, gaining useful usability feedback does not require working software executed on a computer. Prototypes implemented as paper mockups can yield useful usability information. [15][18] These paper mockups are working simulations of the anticipated software.

4.3 Customer collaboration

Customer involvement is essential to building usable software. Customers can play a huge role in gaining access to users for implementing usability engineering practices, especially if the developers do not have a direct connection to a pool of candidate users. Customer cooperation is also important in gaining the flexibility to build the most usable software. Support from the customer is often needed to spend the resources to apply discount usability engineering techniques. Unnecessarily constraining requirements for particular user interface design elements in a contract may preclude options suggested by usability engineering. The discount usability engineering practices do not define a particular customer collaboration model, but customer collaboration is necessary for these practices to succeed.

4.4 Responding to change

To implement discount usability engineering does require planning. Planning is needed to identify users to participate and to coordinate with them to take part in the usability engineering activities. The results of discount usability engineering often reveal differences in assumptions between the developers and the users. For discount usability engineering, responding to change is about how the team adapts to these differences in assumptions identified through user feedback mechanisms. It is impossible to know ahead of time what these issues will be, or what changes will be required to address such issues, and so time and flexibility are both required to address them effectively. Discount usability engineering is not just about responding to change to make software more usable, but it is in fact a catalyst for change by making usability issues more visible throughout the development process.

5 Potential Usability Gaps in Agile Development Techniques

Despite this apparent alignment, usability is rarely mentioned explicitly in discussions of agile development. Constantine writes, "Unfortunately, user-interface design and usability are largely overlooked by the agile processes. With a couple of possible exceptions, users and user interfaces are all but ignored. Informants in the agile process community have confirmed what numerous colleagues and clients have been suggesting all along. XP and the other light methods are light on the user side of software. They seem to be at their best in applications that are not GUI-intensive." [6] There are many areas of interaction in agile development practices in which this lack of explicit usability consideration can lead to usability defects. However, each of these areas may also be viewed as a starting point from which discount usability techniques could be integrated.

5.1 User Feedback

All agile development methods feature user feedback as a key element. XP, for example, calls for an on-site customer to be able to provide regular feedback to the development team. [19] This customer is typically an expert in the functional area the software is addressing. The customer often becomes an expert in the software being developed through regular work with the software development team. This is valuable, but it is difficult for this user to

anticipate the potential usability concerns of different groups of users. For example, the extensive knowledge and understanding the on-site customer has about the system may make it difficult to recognize potential problems that first-time users may have. Similarly, different users may have very different mental models about the area being addressed by the software, and that may require a broader pool of users providing feedback to recognize potential usability concerns. [20] One way to analyze this problem is to consider the value chain of the software being built. [21][22][23] Which links of the value chain actually use the software? Usability defects for a key group in the value chain can undermine the value delivered by the entire chain.

- How can this model of user feedback be adapted to create better insight into the usability of software?
- Can the structured interviewing techniques of discount usability testing improve the quality of the user feedback?
- Are there ways to involve broader participation from the user community in providing usability feedback?

5.2 Acceptance Testing

Another common agile development practice is to define automated acceptance tests. Users play a significant role in this activity by helping understand the expected behavior of the software. These tests are used to verify when a particular portion of work is complete, and to confirm that later versions do not break the feature. However, what if a customer wants to specify that a particular feature, component or system be usable? How would a usability requirement fit into an acceptance testing approach? While these acceptance tests define the external behavior of the software, it is very hard (if not impossible) to devise acceptance tests that reflect the human understanding that underlies the usability of the software. For all but the most basic heuristics, usability testing requires human participants. Understand usability from the perspective of novices means finding new participants for each round of testing. It has been suggested that usability testing can be included as a separate part of acceptance testing. [4] This might work for software applications for small communities of users, all of whom are in regular contact with the development team. However, for cases in which

there is a broader user community, there are several problems with this approach. If you were to conduct usability testing at the end of the development process, you run the risk of having insufficient time and resources to respond to the usability issues raised in the testing. If you do the usability acceptance tests early in the process, then you run the risk of introducing usability defects in later iterations, because there is no regression usability capability as there is for feature validation. If you were to attempt to carry out usability tests as frequently as feature acceptance tests, your budget would balloon out of control.

- Is there a role for usability in agile development acceptance tests?
- At what frequency should usability be reassessed?
- Are new testing tools needed to include some aspect of usability testing in automated acceptance tests?
- Is a new class of tests needed in agile development to address usability?

5.3 Style Guides

Style guides are frequently used in agile software development (as well as in software engineering in general) as vehicles for communicating “best practices.” Typically, this takes the form of guides focused on the programming itself, and not the user interface. [24] Some organizations develop user interface style guides to establish a common look and feel across a wide range of applications. Apple has had a number of such guides, the most recent being their Aqua interface guidelines. [25] These kinds of user interface style guides reflect a good deal of effective usability knowledge, but they are very detailed for a particular user interface style. Most organizations do not already have such a well-defined guide, and it is unlikely that an agile development project would be expected to create such a guide as part of its system development. However, there should be room for heuristic-based usability guides that could be broadly applied across different agile development projects.

- Can usability considerations be incorporated into style guides without weighing them down in unnecessary complexity and detail?
- Would it be possible to create a usability guide for agile development that could be used not just by a team

across projects, but broadly across the agile development community?

5.4 Patterns

Patterns are used as a part of many agile methods. Of course the most widely used and cited patterns focus on design decisions. [26] There are a handful of patterns that address user interface decisions to improve usability, the languages are not as well developed as they are for design, and are rarely, if ever mentioned in the agile development community. [27][28][29][31][30][32][33][34][35] However, patterns are a good tool for helping software engineering make better usability decisions.

While usability heuristics are useful, they are not applied by rote, and they can require interpretation. One typical heuristic that the software should avoid “modality.” That is, the interface should avoid situations where it behaves one way in one mode and another way in a different mode. The motivation for this heuristic is that the more modes a user encounters, the more complex the user's model has to be to understand the system. However, another heuristic could suggest that the paths for novice and expert users could be different. Novices will require more information to help them understand and make decisions in the system. Experts need an interface that optimize how quickly and efficiently they can complete their tasks. This suggests that the system might need a novice and an expert mode. This example illustrates the competing forces that drive user interface design decisions. Just as design patterns capture the tradeoffs of software design decisions, user interface patterns can help developers make better choices about tradeoffs that affect usability. [31]

- Is further development of user interface patterns needed to make them useful for agile developers?
- Are the pattern languages for usability sufficiently developed and packaged to aid the decisions of agile developers?
- How can these languages be best incorporated into the vocabulary of agile developers?

6 One Possible Solution: Discount Usability Engineering with Scrum

There may be other approaches to addressing the challenge I have presented, but one method that I believe warrants particular consideration is Scrum. Scrum is focused on management

instead of engineering practices. Scrum, and is very broad with respect to the kinds of work it organizes. Any work item can be described as a task to be placed in a product or sprint backlog. [36] XP has used in combination with Scrum to provide specific software engineering practices. [37]. Similarly, discount usability engineering could provide the specific usability engineering practices for a Scrum-managed project. There are still many issues that would have to be addressed to make this combination effective, however, I believe the combination is feasible.

7 Conclusion

For many customers and domains, agile software development techniques have improved customer experiences. However, these techniques have not directly or explicitly incorporated the knowledge of usability engineering, even though it is valuable for customers to have usable software and there are discount usability engineering techniques that are similar in spirit to the practices of agile software development. The agile development community should take up the challenge of finding ways to adapt these methods. An examination of some common agile development practices suggests that while they have gaps regarding usability, there are also useful starting points for considering how to incorporate appropriate discount usability engineering methods. One overall strategy that should be feasible is to fashion a discount usability engineering approach for use with Scrum. By addressing these challenges the agile community can build better, more valuable software.

8 Acknowledgements

I would like to thank Jim McCusker, Tim Ruppert, Bill Wake and the Agile Development Conference reviewers for their feedback on this paper.

9 References

- [1] “Agile Alliance Article Page,” <http://www.agilealliance.com/articles/index>, as viewed on January 31st, 2003.
- [2] L. Constantine, “Methodological Agility,” *Software Development*, Vol. 9, No.6, June 2001.
- [3] L. Constantine, “Process Agility and Software Usability: Toward Lightweight Usage-Centered Design,” *Information Age*, August 2002. A white paper reprint of this article can be found at

- <http://www.foruse.com/articles/agiledesign.htm>.
- [4] "Usability And User Interface Design In XP," White paper, July 2002, <http://www.ccpace.com/Resources/UsabilityinXP.doc>
- [5] S. Mitchell, B. Auernheimer, D. Noble, "Scenarios, Tall Tales, and Stories: Extreme Programming the Oak Grove Way," *XP Universe 2001*.
- [6] L. Constantine, Agile Usage-Centered Design, *forUse Newsletter*, No. 12, April 2001, <http://www.foruse.com/newsletter/foruse12.htm>.
- [7] J. Nielsen, *Usability Engineering*, AP Professional, Mountain View, CA, 1993.
- [8] J. Nielsen, "Usability engineering at a discount." In Salvendy, G., and Smith, M.J. (Eds.), *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*, Elsevier Science Publishers, Amsterdam. 394-401, 1989.
- [9] Agile Development Conference Call for Papers, (Web Site: <http://www.agiledevelopmentconference.com/submissions/index.html>).
- [10] J. Nielsen, "Big paybacks from 'discount' usability engineering," *IEEE Software* 7, 3 (May), 107-108.
- [11] J. Nielsen, "Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier," (http://www.useit.com/papers/guerrilla_hci.html) useit.com, 1994, as viewed on January 31st, 2003.
- [12] J. Nielsen, D. Sano, "1994 Design of SunWeb - Sun Microsystems' Intranet," (<http://www.useit.com/papers/sunweb/>) useit.com, 1994, as viewed on January 31st, 2003.
- [13] J. Nielsen, "Heuristic Evaluation," (<http://www.useit.com/papers/heuristic/>) useit.com, as viewed on January 31st, 2003.
- [14] L. Constantine, "Cutting Corners: Shortcuts in Model-Driven Web Design," White paper, 1999, <http://www.foruse.com/articles/shortcuts.htm>.
- [15] J. Nielsen, "Paper versus computer implementations as mockup scenarios for heuristic evaluation," *Proc. INTERACT'90 3rd IFIP Conf. Human-Computer Interaction* (Cambridge, U.K., 27-31 August), 315-320.
- [16] U. Nikula, J. Sajaniemi, "BaSyRE: A Lightweight Combination of Proven RE Techniques," *Proceedings of the International Workshop on Time Constrained Requirements Engineering*, September 9th-13th, 2002.
- [17] "Agile Manifesto" (<http://www.agilemanifesto.org>) as viewed on January 31st, 2003.
- [18] C. Snyder, *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*, Morgan Kaufmann, 2003.
- [19] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 1999.
- [20] J. Preece, Y. Rogers, H. Sharp, *Interaction Design: Beyond Human-Computer Interaction*, Wiley 2002.
- [21] M. Porter, *Competitive Advantage: Creating and Sustaining Superior Advantage: With A New Introduction*. New York: The Free Press, 1998. (Originally published in 1985)
- [22] J. Magretta, "The Power of Virtual Integration: An Interview with Dell Computer's Michael Dell." *Harvard Business Review*, March-April 1998, v76, n2, pp. 72-84.
- [23] D. Dikel, D. Kane, J. Wilson, *Software Architecture: Organizational Principles and Patterns*, Prentice-Hall 2000.
- [24] S. Ambler, "Writing Robust Java Code: The AmbySoft Inc. Coding Standards for Java," v17.01d, January 2000, <http://www.ambysoft.com/javaCodingStandards.html>.
- [25] Introduction to the Aqua Human Interface Guidelines (Web Site: <http://developer.apple.com/techpubs/macosex/Essentials/AquaHIGuidelines>)
- [26] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns*, Reading, MA: Addison-Wesley, 1994.
- [27] Mahemoff, M.J. and Johnston, L.J. "Pattern Languages for Usability: An Investigation of Alternative Approaches." In Tanaka, J. (Ed.), *Asia-Pacific Conference on Human Computer Interaction (APCHI) 98 Proceedings*, 25-31. Los Alamitos, CA: IEEE Computer Society. [In Shonan Village, Japan, July 15-17, 1998]

- [28] Mahemoff, M.J. and Johnston, L.J. (1998). "Principles for a Usability-Oriented Pattern Language." In Calder, P. and Thomas, B. (Eds), *OZCHI '98 Proceedings*, 132-139. Los Alamitos, CA. [In Adelaide, Australia, November 30 to December 4, 1998]
- [29] Mahemoff, M. J. and Johnston, L. J. (1999). "The Planet Pattern Language for Software Internationalisation." In Manolescu, D., and Wolf, B. (Eds.), *Pattern Languages of Program Design (PLOP) 1999*. [In Monticello, IL, September 15-18, 1999]
- [30] J. Tidwell, "A Pattern Language for Human-Computer Interface Design," http://www.mit.edu/~jtidwell/interaction_patterns.html. Originally appeared in *Pattern Languages of Programming 1998*, Washington University Technical Report TR 98-25. Another version appears at <http://time-tripper.com/uipatterns>.
- [31] K. Perzel, D. Kane, "Usability Patterns for Applications on the World Wide Web," *Pattern Languages of Programming 1999*, August 1999.
- [32] J. Borchers, *A Pattern Approach to Interaction Design*, Wiley 2001.
- [33] K. Hornbæk, B. Bederson, C., "Navigation patterns and usability of zoomable user interfaces with and without an overview," *ACM Transactions on Computer-Human Interaction (TOCHI)*, v9,i4, Dec 2002.
- [34] M. van Weile, "Interaction Design Patterns" <http://www.welie.com/patterns/index.html>, as viewed on January 31st, 2003, last updated December 9th, 2002.
- [35] "The Brighton Usability Pattern Collection," <http://www.cmis.brighton.ac.uk/research/patterns/home.html>
- [36] K. Schwaber, M. Beedle, *Agile Software Development with Scrum*. Prentice-Hall, 2002.
- [37] xP@Scrum (Web Site: <http://www.controlchaos.com/xpScrum.htm>)