

Integrating scenario-based usability engineering and agile software development

Jason Chong Lee

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Committee:

D. Scott McCrickard, Chair

James D. Arthur

Robert L. Biddle

Shawn A. Bohner

Manuel A. Pérez-Quñones

March 31, 2010

Blacksburg, VA

Keywords: Agile Software Development, Software Engineering, Agile Usability, Usability Engineering, Central Design Record, Scenario-based Design

Copyright 2010, Jason Chong Lee

ALL RIGHTS RESERVED

Integrating scenario-based usability engineering and agile software development

Jason Chong Lee

ABSTRACT

Agile development methodologies are becoming increasingly popular because they address many risks of software development through things such as quick delivery of working software and responsiveness to change. As agile organizations have begun to develop more user interface-intensive systems, they understand the value and need to design more usable systems. The fields of usability engineering and human-computer interaction are focused on exploring how people interact with computer systems. However, much of this work is inaccessible to agile practitioners because it does not align with core agile values and because there has not been adequate transfer of knowledge between practice and academia.

This motivated my creation of the eXtreme Scenario-Based Design (XSBD) process, an integrated agile usability approach. XSBD provides key usability benefits of the scenario-based design (SBD) approach (an established usability engineering process) and is compatible with an agile development framework modeled on leading agile processes like XP and Scrum. XSBD was designed for use in projects in which a large part of the overall system quality is determined by system usability. This requires close communication and coordination of the disparate usability and agile development work practices. A core aspect of XSBD is the central design record (CDR), which is the shared design representation that guides usability design. It tightly couples usability evaluation results to design features and high level project goals, allowing the usability engineer to leverage key benefits of traditional SBD while working in an agile framework.

I began developing XSBD at Virginia Tech, evaluating it through several student-led development efforts. To improve and demonstrate the applicability of XSBD in practice, I partnered with Meridium, Inc., a software and services company. Using an action research case study method, I worked with several development teams there who used XSBD to develop products. This directly linked usability and HCI research to practice, allowing me to demonstrate XSBD's utility in practice while evaluating it from a theoretical perspective. The results of this work suggest several avenues for further work both to increase its adoption in practice and to link to existing HCI research efforts such as design rationale and knowledge reuse.

To my mom, dad, sister and Stacy

– for all the love and support

Acknowledgements

It has been a long and interesting journey getting to this point and I could not have done it with the guidance and support of many people.

Special thanks go out to my family for their patience and for all their love and support. Mom and Dad, I know I promised to get at least 2 PhDs when I was younger but I'll need a little break before I start on the next one. Eunice, thanks for all the encouragement and for those occasional care packages that kept me from starving.

My thanks go out to my advisor, Dr. D. Scott McCrickard, for directing this research effort to completion. I could not have done it without your guidance, your insights and your friendship. Thank you for the late-night paper writing sessions, the lab 'battles', the steady stream of sodas and snacks, and for convincing me that I could do this.

I'd also like to thank my committee members: Dr. Robert Biddle, for your valuable insights into agile methods and for your help in developing my evaluation approach. Dr. James D. Arthur, for all our discussions about software vs usability and helping make sure I didn't miss the forest for the trees. Dr. Shawn Bohner, for your support and guidance when I was struggling to find funding for this work. Dr. Manuel Pérez-Quiñones, for helping make sure I was asking the right questions.

My thanks go out to my colleagues in our lab at Virginia Tech and the CHCI. Stacy Branham, for all the love and support and for helping me see things from a different perspective. Shahtab Wahid, for working with me on all those papers, projects and grants...I'll always remember LINK-UP. Christa Chewar, for starting me on this path. Laurian Vega, for keeping me from crawling into my shell. Tejinder Judge, for pointing me to what I really wanted. Thanks also to all those that participated in my early studies and to the faculty and staff at the CHCI and the computer science department.

I'd also like to thank those at Meridium for helping make this work possible. Hari Pulijal, for your guidance and support throughout this endeavor. The grant would not have been possible without your support. Sarav Bhatia, John Renick, Dr. Todd Stevens, and everyone else that was involved in the case study projects, for your valuable feedback, and support.

And thanks to the many other friends and colleagues who helped me along the way: Mishu, Ricky, Pardha, Ben, Edwin, Justin, Manas, Ahmed, John, Josh, Dong Kwan, Seon Ho, Kibum, Miten, Maria, Travis, Saurabh, Ali, Jeremy, Greg, Sirong, Priya, Shannon... (I'm sorry to anyone I left out)

Also, thanks to the NSF for their support. This work was partially funded by a National Science Foundation Small Business Technology Transfer Phase I Grant No. #0740827.

Table of Contents

| | | |
|----------|--|-----------|
| 1 | INTRODUCTION..... | 1 |
| 1.1 | CHALLENGES FACING AN INTEGRATED APPROACH..... | 3 |
| 1.2 | THESIS STATEMENT..... | 4 |
| 1.3 | SOLUTION APPROACH: XSBD | 4 |
| 1.4 | CONTRIBUTIONS..... | 6 |
| 2 | RELATED WORK | 8 |
| 2.1 | USABILITY ENGINEERING | 8 |
| 2.1.1 | <i>Phases of usability engineering</i> | <i>9</i> |
| 2.1.2 | <i>Scenario-based design</i> | <i>11</i> |
| 2.2 | EMERGENCE OF AGILE METHODS | 13 |
| 2.2.1 | <i>The Agile Manifesto.....</i> | <i>13</i> |
| 2.2.2 | <i>Agile landscape.....</i> | <i>15</i> |
| 2.2.3 | <i>Extreme programming</i> | <i>15</i> |
| 2.2.4 | <i>Scrum.....</i> | <i>17</i> |
| 2.3 | ENABLING AGILE USABILITY | 18 |
| 2.3.1 | <i>Convergence points between agile and usability.....</i> | <i>19</i> |
| 2.3.2 | <i>Divergence points between agile and usability.....</i> | <i>20</i> |
| 2.4 | CHAPTER SUMMARY | 25 |
| 3 | BACKGROUND WORK | 26 |
| 3.1 | LEVERAGING A DESIGN REPRESENTATION IN LINK-UP..... | 26 |
| 3.1.1 | <i>Problem overview and research goal</i> | <i>27</i> |
| 3.1.2 | <i>Background.....</i> | <i>27</i> |
| 3.1.3 | <i>Analytic module development</i> | <i>29</i> |
| 3.1.4 | <i>Pilot testing evaluation methods.....</i> | <i>32</i> |
| 3.1.5 | <i>Summary of study findings.....</i> | <i>33</i> |
| 3.2 | BRINGING DESIGN REPRESENTATIONS TO THE FOREFRONT..... | 33 |
| 3.2.1 | <i>Problem overview and research goal</i> | <i>34</i> |
| 3.2.2 | <i>Background.....</i> | <i>35</i> |
| 3.2.3 | <i>Enhancing LINK-UP.....</i> | <i>36</i> |
| 3.2.4 | <i>User study description</i> | <i>36</i> |
| 3.2.5 | <i>Summary of study findings.....</i> | <i>37</i> |
| 3.3 | OTHER PRELIMINARY WORK | 38 |

| | | |
|----------|---|-----------|
| 3.4 | CHAPTER SUMMARY | 39 |
| 4 | DEVELOPING XSBD AROUND THE CDR..... | 40 |
| 4.1 | DEFINING THE CENTRAL DESIGN RECORD IN LINK-UP | 40 |
| 4.1.1 | <i>Problem overview and research goal</i> | <i>41</i> |
| 4.1.2 | <i>Motivation and related work.....</i> | <i>41</i> |
| 4.1.3 | <i>LINK-UP.....</i> | <i>42</i> |
| 4.1.4 | <i>The claims library.....</i> | <i>44</i> |
| 4.1.5 | <i>Requirements analysis module.....</i> | <i>44</i> |
| 4.1.6 | <i>Central design record module.....</i> | <i>45</i> |
| 4.1.7 | <i>Case study discussion</i> | <i>47</i> |
| 4.1.8 | <i>Summary of study findings</i> | <i>49</i> |
| 4.2 | DEVELOPING THE XSBD PROCESS | 49 |
| 4.2.1 | <i>Interface architecture design</i> | <i>51</i> |
| 4.2.2 | <i>Collaboration through the CDR</i> | <i>52</i> |
| 4.2.3 | <i>Usability evaluations through the CDR.....</i> | <i>52</i> |
| 4.2.4 | <i>Design case studies.....</i> | <i>53</i> |
| 4.2.5 | <i>Discussion of results</i> | <i>56</i> |
| 4.2.6 | <i>Summary of study findings</i> | <i>58</i> |
| 4.3 | CHAPTER SUMMARY | 59 |
| 5 | EXTREME SCENARIO-BASED DESIGN..... | 60 |
| 5.1 | OVERVIEW OF XSBD PROCESS | 61 |
| 5.1.1 | <i>XSBD in brief.....</i> | <i>61</i> |
| 5.1.2 | <i>Characteristics of an XSBD project.....</i> | <i>62</i> |
| 5.2 | ROLES IN THE XSBD PROCESS..... | 62 |
| 5.3 | THE CENTRAL DESIGN RECORD..... | 65 |
| 5.3.1 | <i>Overview of the CDR</i> | <i>65</i> |
| 5.3.2 | <i>Project vision and prioritized project goals.....</i> | <i>66</i> |
| 5.3.3 | <i>Design artifacts.....</i> | <i>67</i> |
| 5.3.4 | <i>User testing results</i> | <i>70</i> |
| 5.4 | MAINTAINING SYNCHRONICITY THROUGH PLANNING AND PROCESSES | 73 |
| 5.4.1 | <i>XSBD process description.....</i> | <i>73</i> |
| 5.4.2 | <i>Synchronization points.....</i> | <i>78</i> |
| 5.5 | BALANCED DECISION MAKING | 79 |
| 5.5.1 | <i>Balancing different concerns</i> | <i>79</i> |
| 5.5.2 | <i>Claims to compare design features and resolve disagreements.....</i> | <i>81</i> |
| 5.6 | CHAPTER SUMMARY | 81 |

| | | |
|----------|---|------------|
| 6 | SUBSTANTIATING XSBD IN PRACTICE | 82 |
| 6.1 | SUBSTANTIATION APPROACH | 83 |
| 6.1.1 | <i>Action research</i> | 84 |
| 6.1.2 | <i>Data collection</i> | 85 |
| 6.2 | COLLABORATION WITH MERIDIUM | 87 |
| 6.3 | FIRST CASE STUDY: WEB COMPONENTS..... | 88 |
| 6.3.1 | <i>Project personnel</i> | 89 |
| 6.3.2 | <i>Project description</i> | 89 |
| 6.3.3 | <i>Discussion of Web Components project</i> | 101 |
| 6.4 | SECOND CASE STUDY: TOUCH SCREEN | 103 |
| 6.4.1 | <i>Project personnel</i> | 104 |
| 6.4.2 | <i>Project description</i> | 105 |
| 6.4.3 | <i>Discussion of Touch Screen project</i> | 131 |
| 6.5 | OVERALL DISCUSSION | 134 |
| 6.6 | CHAPTER SUMMARY | 137 |
| 7 | PRINCIPLE-BASED ANALYTIC EVALUATION OF XSBD..... | 138 |
| 7.1 | EVALUATING WHETHER XSBD IS AN AGILE PROCESS | 139 |
| 7.1.1 | <i>Individuals and interactions over processes and tools</i> | 140 |
| 7.1.2 | <i>Working software over comprehensive documentation</i> | 141 |
| 7.1.3 | <i>Customer collaboration over contract negotiation</i> | 142 |
| 7.1.4 | <i>Responding to change over following a plan</i> | 143 |
| 7.2 | EVALUATING WHETHER XSBD IS A USABILITY ENGINEERING PROCESS..... | 143 |
| 7.2.1 | <i>Requirements analysis</i> | 144 |
| 7.2.2 | <i>Design</i> | 147 |
| 7.2.3 | <i>Evaluations</i> | 149 |
| 7.3 | ADDRESSING THE CONFLICTS BETWEEN USABILITY AND AGILITY | 150 |
| 7.3.1 | <i>Incremental development vs phased development</i> | 150 |
| 7.3.2 | <i>Working software vs design documentation</i> | 152 |
| 7.3.3 | <i>Customer focus vs end user focus</i> | 153 |
| 7.3.4 | <i>Test driven development vs usability test driven development</i> | 155 |
| 7.3.5 | <i>Shared understanding vs distinct roles</i> | 157 |
| 7.3.6 | <i>Simplicity in the code vs simplicity in the UI</i> | 158 |
| 7.4 | VALUES AND PRINCIPLES OF A COMBINED AGILE USABILITY APPROACH..... | 160 |
| 7.4.1 | <i>New principles</i> | 160 |
| 7.4.2 | <i>Modified principles</i> | 161 |
| 7.4.3 | <i>Unchanged principles</i> | 162 |

| | | |
|--------------------|--|------------|
| 7.5 | CHAPTER SUMMARY | 164 |
| 8 | CONCLUSIONS AND FUTURE WORK | 165 |
| 8.1 | KEY FINDINGS | 166 |
| 8.2 | CONTRIBUTIONS OF THIS RESEARCH EFFORT | 170 |
| 8.2.1 | <i>Contributions to the agile community</i> | 170 |
| 8.2.2 | <i>Contributions to the research community</i> | 172 |
| 8.3 | FUTURE WORK | 172 |
| 8.4 | SUMMARY | 174 |
| 9 | REFERENCES..... | 176 |
| APPENDIX A. | CRITICAL INCIDENTS FROM CASE STUDIES | 183 |
| APPENDIX B. | RETROSPECTIVES FROM CASE STUDIES..... | 260 |
| APPENDIX C. | CASE STUDY INTERVIEWS..... | 265 |
| APPENDIX D. | IRB INFORMED CONSENT AND APPROVAL..... | 334 |

List of Figures

Figure 1. Overview of solution approach. 4

Figure 2. Overview of solution approach. Chapter 2 will provide a summary of related agile and usability work--focusing on key similarities and differences between the two areas. 8

Figure 3. Scenario and associated claim for the notification collage. The claim highlights the tradeoffs of the bulletin board metaphor used in the system. 12

Figure 4. Overview of solution approach. This chapter summarizes my initial work on scenario-based design and design representations which would later inform my work on the CDR and agile usability. 26

Figure 5. Left: early storyboard of the analytic module; Right: Screen from the functional prototype used in pilot testing..... 30

Figure 6. Basic sequence of tasks of the analytic module of LINK-UP..... 30

Figure 7. Norman’s conceptualization of the system image. The designer captures system intentions in a prototype, along with supporting documentation, enabling the study of user interaction with the system..... 35

Figure 8. Overview of solution approach. This chapter summarizes how I developed the CDR and used it as a core feature of the XSBD agile usability approach..... 40

Figure 9. LINK-UP’s iterative design process and CDR development. Designers start at the center identifying requirements and a target IRC goal. Design iterations through the CDR include design claims which are tested through evaluations leading to convergence of the design and user models. 43

Figure 10. Part of the requirements analysis module. The progress/navigation bar is always visible on the right side of the screen to show designers where they are in the process..... 45

Figure 11. Part of the central design record module. Users create a set of scenarios and corresponding claims for each supported activity. Areas of the CDR module can be accessed in any order..... 46

Figure 12. Key steps in XSBD process. Design artifacts are shown in white..... 51

Figure 13. Example claims map. The root concept claim (green) is linked to activity claims (blue) which are linked to implementation claims (yellow). Note the progressive growth through the iterations. (Note that only claim features are included in the diagrams)..... 54

Figure 14. Overview of solution approach. Chapter 5 will provide a summary of the XSBD process as it currently exists. Details of how it was used in practice will be described in Chapter 6. 60

Figure 15. Streamlined version of the Central Design Record developed for use in the XSBD process. 65

Figure 16. Example scenario. 68

| | |
|--|-----|
| Figure 17. Medium-fidelity prototypes provide more details than a low-fidelity prototype but does not provide exact positioning/color information. | 69 |
| Figure 18. Example claim. | 70 |
| Figure 19. Relationship between design goal, design claim and testing result for the automated checkout system. | 72 |
| Figure 20. The basic handoff structure between software and usability engineers. | 74 |
| Figure 21. Parallel development and usability tracks for a complete release cycle. | 74 |
| Figure 22. Summary of usability testing results and proposed solution. | 80 |
| Figure 23. Overview of solution approach. Chapter 6 will summarize how the XSBD process was substantiated and refined using an action research case study approach. | 82 |
| Figure 24. Stages of action research [124] | 84 |
| Figure 25. Sample online critical incident form. | 86 |
| Figure 26. High level goals for Web Components project. | 92 |
| Figure 27. Metrics for Web Components project. | 92 |
| Figure 28. Image A shows that development and usability are synchronized. Image B shows usability work being delivered late and being disregarded by development as a result. | 95 |
| Figure 29. Key design claims maintained by the usability engineer. | 96 |
| Figure 30. The workday split into three eight hour slices. Onsite and offsite work proceeds asynchronously so issues cannot be addressed until at least the end of a 24-hour cycle. | 97 |
| Figure 31. High level goals and metrics for Touch Screen project. | 107 |
| Figure 32. Conceptual image of how the usability engineer communicated results. Each design claim (middle) is linked to a high-level goal (left) and is validated through usability tests (right). | 109 |
| Figure 33. Document summarizing usability work. Contains high-level goals, design claims, usability results and links to other resources. | 113 |
| Figure 34. Design mockup with associated claims. Downsides are shown in pink while upsides are shown in green. Claim relationships to high-level goals are shown using color coded dots. | 114 |
| Figure 35. Planned work synchronization based on XSBD process. | 116 |
| Figure 36. Actual work synchronization. Note that horizontal widths of iterations vary only to better show contents of each. Each iteration covered the same amount of time. | 117 |
| Figure 37. Claim describing start page logo. Downsides are shown on the left and upsides are on the right. | 122 |
| Figure 38. Usability evaluation results showing the design recommendation (description), evidence from the study (study motivation), claim tradeoffs (prediction errors) and related high-level goals (goals motivation). | 124 |

Figure 39. Overview of solution approach. Chapter 7 will include an analytic analysis of the XSBD approach with respect to key agile and usability values/principles..... 138

Figure 40. Overview of solution approach. Chapter 8 will summarize key findings of this research effort and directions for future work. 165

List of Tables

Table 1. Summary of strengths and weaknesses of LINK-UP derived from three development case studies..... 47

Table 2. Table shows how the XSBD approach addresses tensions between agile methods and usability. 57

1 Introduction

Emergence of agility. Software engineering is a relatively new area compared to other engineering disciplines [108]. This fact, combined with the increasing ubiquity and complexity of software-based systems and how they are integrated into modern society have greatly increased the difficulty of engineering and delivering software that is on-time, on-budget and satisfies the requirements of its users [12, 19, 128, 129]. Agile methodologies such as extreme programming (XP) and Scrum, have emerged in the last decade as a way to address these risks by focusing heavily on quick delivery of working software, incremental releases, team communication, collaboration and the ability to respond to change [5, 11, 12, 36, 45, 53, 59, 69, 115, 116]. A significant majority of practitioners who have been on agile teams indicate that they produce higher quality software, greater productivity and higher stakeholder satisfaction [1]. As such agile methods in one form or another have become increasingly popular in practice and are being used to develop an increasing range of software systems [17, 119, 120, 121].

However, agile development methodologies—which focus on the design, implementation and maintenance of the software, did not originally focus on developing systems that were usable [4, 56, 102]. This was because early agile projects did not require the development of complex, interactive user interfaces and because there was an assumption that close collaboration with customers would result in usable systems. However, as agile methods were used while developing more UI-intensive applications, practitioners found the resulting user interfaces to have poor usability even though they met functional and performance requirements.

The need for usability. Usability is closely related to Human-Computer Interaction (HCI)—the area of study that focuses on understanding how people use computer devices, and how such devices can be made useful and usable [22]. It bridges the social and behavioral sciences with the engineering of computer-based systems and is thus an inherently multidisciplinary endeavor. Usability engineering is specifically concerned with the practical application of knowledge about how people use systems to develop interfaces that can be used efficiently and effectively. It deals with issues such as system learnability, efficiency, memorability, errors and user satisfaction [38, 54, 95, 107, 111]. Usability engineering can give insights into user motivations, characteristics and work environments to further HCI research. Much of this work in HCI and usability is done in academia and research labs where it is not easily accessible to agile practitioners [23, 26, 27, 60, 70, 101, 110, 135]. If this theory-practice gap between usability/HCI research and agile methods in practice is closed then agile practitioners can

benefit from practical usability methods compatible with agility while usability researchers can benefit from real-world data and experience that can further their research efforts.

Integrating usability and agility. There is a need to find ways to integrate usability practices into agile methodologies in such a way that the core benefits of agility—fast delivery of working software, adherence to customer goals and responsiveness to change—are preserved. Agile practitioners are unlikely to accept usability methodologies that are not closely aligned with core agile values and principles [13, 36, 90]. Because of this, existing work on integrating usability into software engineering methods are not sufficient [10, 103, 109, 122]. By understanding core convergence and divergence points between agility and usability, one will be able to better understand the key challenges to integrating the two areas and also where similarities can be exploited to facilitate that integration.

Agile practitioners and researchers have acknowledged the need to develop systems that meet usability requirements in addition to meeting functional and market requirements and have begun to explore ways of incorporating usability into agile methods [3, 4, 14, 30, 34, 43, 46, 80, 83, 84, 85, 97, 102, 117, 127]. Some have argued for an agile-centric approach where existing agile teams learn about and integrate usability practices into their day to day tasks [83, 102]. One potential problem with this approach is that effective UI design can be difficult for complex systems with a heavy emphasis on usability. They may require a level of expertise and knowledge that cannot be learned in a short period of time. Others such as Cooper advocate a usability-centric approach where usability professionals first interact with customers to collect data from end users and develop the UI design before working with developers to implement it using a traditional agile approach [34, 90]. However, this approach runs counter to agile practice of continuous development and feedback and would be a waste of resources as developers would have to wait until the UI design was ready [90]. One issue to address is where an integrated approach should lie along these two extremes.

Within an integrated approach, there is also the question of how appropriate tradeoffs between usability and development concerns are handled. With development-centric approaches such as that advocated by Patton, there is an inherent conflict of interest that can arise if a single person is responsible for both the UI design and development of a feature. Given a limited amount of time to complete a feature, the developer is more likely to sacrifice usability to get the code implemented since functioning code is central to agile methods [13]. The problem of how to handle tradeoffs is also critical with the more common integration approach of having separate usability and development

teams work in parallel [84, 97, 127]. Different stakeholders such as software developers and usability engineers and different backgrounds, focus points and concerns which can result in conflicts between the two groups [73]. In addition, parallel design and development efforts run the constant risk of design drift where implementations do not match up with original designs [30, 51].

The integration challenges described above form the guiding framework of my dissertation work. They are summarized below:

1.1 Challenges facing an integrated approach

The successful integration of usability engineering and agile software development practices depends on a number of different factors that have been identified and addressed over the course of this research effort.

1. **Comparison of core principles.** In considering usability methods and agile methods, what are the convergence and divergence points from a high-level principles perspective? It is important to identify where difficulties lie in integrating the two approaches, where similarities can be exploited in a combined approach and what the costs and benefits of the resulting approach are.
2. **Balance of power.** In developing a specific instantiation of a combined approach, how should the approach be balanced between usability and agile methods? For instance, an agile-centric approach may involve providing some level of usability-training to agile developers while a usability-centric approach may have usability engineers complete UI designs before agile developers begin their work. The integration approach will have an impact on the long-term viability and acceptance of an integrated process in practice.
3. **Checks and balances.** How can appropriate tradeoffs between the agile and usability practices in the integrated approach be made? Any integrated agile usability approach will have tradeoffs that may, for example, sacrifice development velocity for validation of the usability of a user interface. The integrated approach should help agile usability teams make these tradeoffs given the specific needs of their development project.
4. **Synchronicity.** How can the agile development and usability engineering sides of a project stay synchronized throughout the development effort? This is critical as design drift between the interface design and its implementation can render the usability work irrelevant and result in a system that has poor usability. This problem is exacerbated by the speed at which agile development projects proceed. Addressing this issue depends on proper development and use of shared design representations, communication practices and process planning structures within the team.

1.2 Thesis statement

The goal of my research efforts is to develop and demonstrate the utility of the eXtreme Scenario-Based Design process (XSBD) in integrating usability engineers into an agile development team to support the efficient development of usable software systems. The analysis, development and substantiation of this approach and how it addresses the key integration challenges form the body of this research and can be summarized in the following statement:

The successful integration of scenario-based design processes into an agile development framework—XSBD—requires the development of appropriate communication, process planning and design artifact sharing techniques to enable proper synchronization and balanced tradeoffs between usability and agile development concerns—refined and evaluated through real world development efforts.

1.3 Solution approach: XSBD

Addressing the thesis statement and the integration challenges required a multi-stage approach summarized in Figure 1.

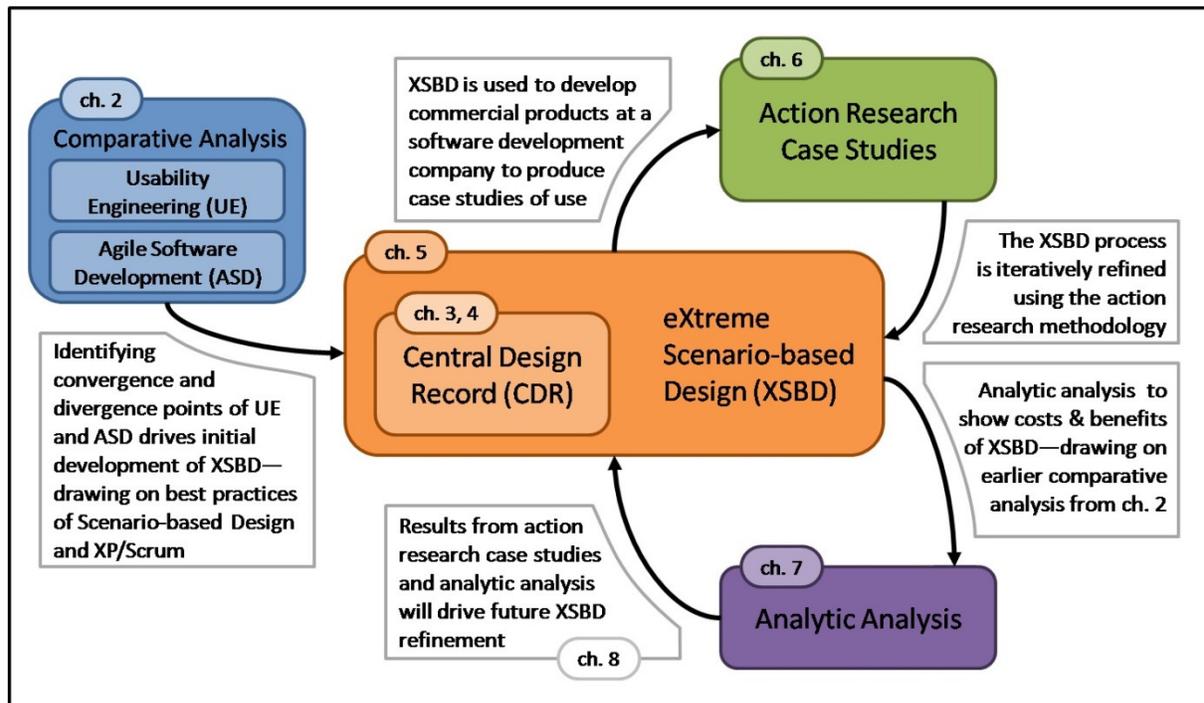


Figure 1. Overview of solution approach.

Agile software development is centered on the Agile Manifesto, a collection of core values and principles that was developed by leading agile practitioners to provide a guiding framework for the development of agile methods [13]. Similarly, usability engineering methods share many common

values and guiding principles [38, 54, 95, 107, 111]. A comparison of both disciplines at a conceptual level, towards identifying where they are in agreement and where they conflict, provided guidance as to what potential problems an integrated approach would encounter. This initial analysis, described in Chapter 2, formed the theoretical basis of my integrated approach and directed later evaluation efforts.

The XSBD approach provides key usability benefits of the scenario-based design approach—an established usability engineering process developed by Carroll and Rosson—and is compatible with an agile development framework modeled on leading agile processes like XP and Scrum [11, 12, 115, 116]. The XSBD approach was designed for use in projects in which a large part of the overall quality of the system will be determined by how usable it is. Software development and usability design proceeds in parallel, which allows the dedicated usability and development personnel to work together quickly and efficiently. This requires close communication and careful coordination of the disparate usability and agile development work practices. A core aspect of XSBD is the central design record (CDR), which is the shared design representation that guides usability development [64, 66, 67]. It tightly couples usability evaluation results to design features and high-level project goals, allowing the usability engineer to leverage key practices (and the resulting benefits) of SBD while working in an agile incremental development cycle. The CDR is grounded in a planning and development process that integrates software development and usability engineering efforts. My preliminary work in using design representations in usability engineering processes is described in Chapter 3. Based on this initial work, I developed the CDR concept and initial tests of the representation in student-led agile usability development teams at Virginia Tech. This work is described in Chapter 4.

Since agile methods are mostly used by software developers in industry, I determined that the optimal way to substantiate the utility of the XSBD approach was to use it in practice at a software development firm. By partnering with Meridium, Inc., a software and services company based in Roanoke, Virginia, I was able to have several development teams use the approach to develop commercial products. I used an action research methodology to study the XSBD approach in practice. In this approach, I was embedded with the development teams at Meridium and continuously refined the process based on observations and feedback from the development team members. This approach is ideal for studying new development methodologies and allowed me to improve the XSBD process while providing practical, actionable feedback to the development teams at Meridium. The XSBD process that resulted from the action research case studies is detailed in Chapter 5. I will then step back and revisit in detail the methodology and evaluation results stemming from those case studies in Chapter 6. Finally, I will present the results of a comprehensive analytic evaluation of the XSBD

process in Chapter 7 in which I demonstrate that XSBD is a valid agile usability approach by showing how it aligns with core values of both agility and usability. In this analysis, I discuss how divergence points between agility and usability were addressed through specific XSBD practices and discuss their tradeoffs with respect to balancing usability and agile concerns. Based on this analysis and the action research case studies, I provide a set of agile usability principles to provide further guidance to agile and usability practitioners in general. Key findings and future work are detailed in Chapter 8.

1.4 Contributions

The overall goal of this project was to develop and promote a well-defined agile usability process that practitioners can use and adapt to their agile organizations. An additional goal is to provide a framework and general guidelines within which agile software developers and usability engineers can work together to efficiently create usable software systems through common practices from both disciplines. In summary, this work has and will benefit the research and agile practitioner communities in the following ways.

Contributions to agile practitioners

- **Detailed description of agile usability approach—XSBD.** XSBD is a specific instantiation of agile usability that has been substantiated in practice. It provides actionable guidance to practitioners who want to integrate usability into their agile organizations.
- **Central Design Record as shared design representation.** Usability engineers can develop and use the CDR to help them use a scenario-based usability design process within an agile framework.
- **Detailed case studies of XSBD use in practice.** Several detailed case studies of XSBD can provide utility not only to those interested in adopting it in practice, but also to agile usability practitioners in general.
- **Agile usability principles.** Generalized guidance to agile practitioners is provided in the form of a set of agile usability principles.
- **Introduction of agile usability to new domain.** The XSBD approach has been used in a number of development efforts at Meridium, Inc., an agile software development organization with limited previous experience with usability processes.

Contributions to the research community

- **Addressing theory-practice gap.** This work addresses the theory-practice gap by bringing together usability research and agile practitioner communities using an action research methodology [23, 60, 70, 101, 110, 135].

- **Contributing to general body of software and usability knowledge.** This work also contributes to the general body of knowledge with software engineering and usability engineering focusing specifically on the challenges of integrating usability into an agile software development framework.
- **Design rationale in practice.** This work contributes to the general research area of design rationale use [20, 25, 27, 47, 70, 126, 130, 131]. It provides a specific example of how design rationale can be used in practice to design usable systems.

2 Related work

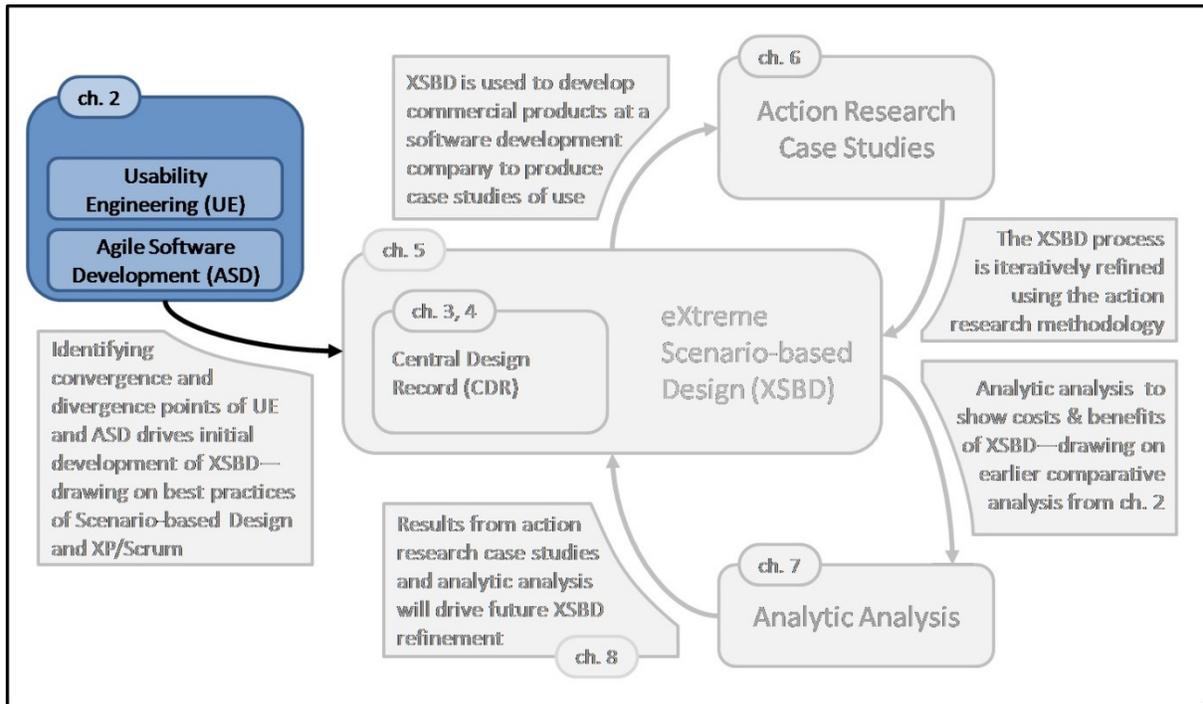


Figure 2. Overview of solution approach. Chapter 2 will provide a summary of related agile and usability work--focusing on key similarities and differences between the two areas.

This chapter presents an overview of literature relevant to this work (See Figure 2). It first introduces key concepts of usability engineering—detailing the major parts of scenario-based design. It then details the core values of agile methodologies—and provides an overview of extreme programming and Scrum—the primary agile methodologies XSBD draws from. It highlights the need for developing ways to integrate usability into agile methodologies and gives an overview of key convergence and divergence points between agility and usability that any integrated approach needs to address. A review of ongoing work in the area of agile usability is provided within the context of these convergence and divergence points.

2.1 Usability engineering

Usability bridges the social and behavioral sciences with the engineering of computer-based systems and is thus an inherently multidisciplinary endeavor [22]. Usability engineering is specifically concerned with the practical application of knowledge about how people use systems to develop interfaces that can be used efficiently and effectively. It deals with issues such as system learnability, efficiency, memorability, errors and user satisfaction [38, 54, 95, 107, 111]. Usability engineering can give insights into user motivations, characteristics and work environments to further HCI research.

In the 1970s and 80s, user interface design became an increasingly important aspect of software development as software became more interactive [111]. Computers became more powerful, more affordable and more ubiquitous as they were used for a broader range of applications. Users became less technical and the inner workings of the computer system were increasingly abstracted away from them. This was especially evident with the rise of graphical user interface (GUI) systems such as the Macintosh and Windows operating systems that leveraged things such as direct input systems and interactive graphical displays. This in turn led to the emergence of usability as a critical quality factor in systems design.

2.1.1 Phases of usability engineering

Usability practitioners and researchers began developing methods and tools to design usable systems. For example, some began leveraging work from psychology and industrial engineering. Card, Moran and Newell focused on performance optimization through simpler displays, reduced keystrokes and greater efficiency [21]. User evaluation techniques such as think-aloud protocols were developed to get feedback from users that could inform subsequent development efforts [48]. These various tools methods came together into usability engineering processes that were developed through the 1980's up to the present.

Usability engineering processes are generally user-centered, in that their goal is to develop a system that meets end user needs by involving them throughout the development process [38, 54, 95, 107, 111]. Usability engineering processes are similar in that they all operate in an iterative loop that involves requirements analysis, design, and evaluation. The output of each iteration is dynamic and subject to change based on the output from the previous step. Thus, the usability goals and requirements might change based on how users reacted to the design [24]. These steps are outlined below:

Requirements analysis. Usability processes rely on a requirements analysis phase to define project goals, gain an understanding of the end users, their work context and their activities [111, 123]. All requirements cannot be defined at the beginning of the process, but a large scale requirements analysis phase at the beginning of a project is common [90]. This phase will often include things such as observations of work practice where usability engineers visit the end user workplace to understand users' day-to-day experiences, and the social and physical context in which the system will be used

[111]. More participatory methods may be used where end users are first observed in their normal work practice and then participate in an active discussion about those activities.

An important early part of usability engineering is setting appropriate high-level design goals to guide subsequent development iterations. These goals are important in that they can help designers prioritize design options and make informed decisions [95]. Newman introduced the concept of *critical parameters*—figures of merit that transcend specific applications and focus on the broader purpose of a particular class of technologies—as one way to define system goals [91, 92]. They can function as benchmarks that provide a direct and manageable measure of how well a design meets its purpose. Critical parameters are useful in that they can be used to reliably compare the quality of different designs. For example, critical parameters for a car could be things such as mileage, maximum speed and acceleration. Within the field of HCI, McCrickard and Chewar have developed critical parameters for notification systems—systems used in dual task situations to supply users with information while minimally interrupting them from some primary task [33, 76, 77]. My early work in usability engineering and agile usability processes focuses on the development of such systems.

Design. During design, usability engineers design the user interfaces that meet user needs identified during requirements analysis. This often begins with defining the activities and workflows that the user needs the system to support. Scenarios are commonly used as a way to provide easily understandable and flexible examples of usage in the users work context [24]. More structured methods such as hierarchical task analyses can also be used to decompose tasks into ordered subtasks. The usability engineer then needs to define how those tasks will be supported through the user interface. This involves determining how information will be delivered to the user and how the user will interact with the system. Specific UI designs are developed through prototyping—where the designer develops models of the design of varying levels of specificity. For example, a prototype could be a simple sketch or some piece of coded functionality. These are used to explore the design space and get feedback from end users and other stakeholders.

Evaluations. Usability evaluations are any analysis or empirical study of the usability of the system [38, 54, 95, 107, 111] and effectively close the loop connection requirements to the design. They provide feedback on whether design is meeting the project requirements, identify usability problems, and can also uncover new requirements. Usability engineering methods leverage a variety of analytic and empirical evaluation methods. Analytic methods detailed analyses or theoretical models of the user interface, usually done by usability experts [111]. Empirical methods such as controlled usability

testing in a laboratory or longitudinal studies of the system in use provide actual feedback from users but can be more time and resource-consuming to run than analytic methods. The actual type of evaluation used will depend on the type of feedback the usability engineer needs and the circumstances of the specific project.

2.1.2 Scenario-based design

There are a number of different usability engineering methodologies and techniques that cover different aspects of the usability engineering lifecycle including requirements analysis, prototyping and empirical evaluation [38, 54, 95, 107, 111]. In my work, I use Carroll and Rosson's scenario-based design (SBD) as an exemplar usability engineering approach that embodies each of the core characteristics described in section 2.1.1. SBD is an established usability engineering approach centered on the use of *scenarios*—narratives describing users engaging in some task, in conjunction with design knowledge components called *claims*, which encapsulate the positive and negative effects of specific design features as a basis for creating interactive systems [24, 26, 111]. Claims provide compact, designer-digestible packets of knowledge ideal for use in time-critical design activities [125, 126]. Figure 3 shows an example scenario and claim for the notification collage, a virtual notice board to allow users to maintain awareness of people they work with [49].

Scenario-based design specifies four design phases: requirements analysis, activity design, information design, and interaction design [111]. *Requirements analysis* is where designers first collect information on current practices through interviews, ethnographic studies and other data gathering techniques. This information is used to construct a root concept document, which describes the overall vision for the system and stakeholder descriptions. The designers then craft problem scenarios and claims to describe how tasks are currently done and what key problems and issues exist. In *activity design*, designers develop scenarios and claims to describe activities and tasks the new system will support based on the previously developed problem scenarios and claims. In *information* and *interaction design*, designers determine how the activities will be supported through the information the interface provides and the interactions it supports. These phases, though defined serially, are done iteratively and often intermingle in practice as the nature of the task-artifact cycle means that requirements and user needs may change as the system is iteratively developed. Each phase is followed by some type of usability evaluation to verify that the design meets the goals defined in requirements analysis and that it is meeting end user needs. Specific evaluation, prototyping and requirements analysis methods are not required and are chosen based on the nature of the system being developed and other situational factors.

Pascal, a graduate student, is working on a paper related to his research. While working on the paper, he also wishes to be informed of information that is being shared within his lab. By using the Notification Collage (NC), which runs on his second monitor, Pascal can now glance at the NC every once in a while in order to see the posted items. When looking at the NC, he visually scans the randomly placed items that are on top. As he looks at the various types of information posted, he gains an understanding of the information contained in the items that are completely visible, but does not know if the information is recent. Knowing that he must find out at a later time when the information items were posted, he returns to his research paper.

Information artifacts haphazardly posted in an unorganized fashion onto a public display for relevant information delivery, similar to postings on a bulletin board.

- + allows users to gain an understanding of an item's age/applicability with respect to the number of items that may be covering it**
- + the lack of information categorization accommodates a wide range of different types of information to be conveyed through the display**
- BUT overlapping items due to the lack of organization can hinder efforts to read/see a particular information item**
- BUT the actual age of an item is not apparent**

Figure 3. Scenario and associated claim for the notification collage. The claim highlights the tradeoffs of the bulletin board metaphor used in the system.

Carroll asserts that theory-grounded HCI research can drive innovation by expressing, testing, and reusing “falsifiable” hypotheses (or *claims*) about the psychological effects an artifact has on a user. Scenario-based design (SBD) is an approach to interface development, providing an inquiry method to help designers reason about elements of a usage situation and receive participatory feedback from stakeholders [26, 27, 111]. Using Carroll’s approach, HCI professionals and software developers conduct an explicit *claims analysis* in formative and summative usability engineering efforts, continuously striving to balance and validate tradeoffs of use. A claims analysis record for a single system, and the accumulation of records from multiple systems, holds valuable design knowledge that, as Carroll has argued, should facilitate component reuse, motivate further empirical research, and inspire high-impact innovative development [26]. Furthermore, combined with the use of critical parameters, the SBD approach can help practitioners by driving the development of demonstrably more useable systems [91, 92].

My work draws primarily from scenario-based design as an exemplar usability engineer process that embodies many of the key values and practices common among usability engineering processes. SBD

is also used as it was developed to bridge the apparent gap between academic research in HCI and usability engineering in practice.

2.2 Emergence of agile methods

In software engineering, prescriptive process models are used to control the unpredictability and fluidity that can surround software development. Traditional software methodologies adopt a ‘divide and conquer’ approach to development with different teams working on different phases of development such as requirements analysis, implementation and testing. For example, the waterfall model of development, one of the oldest software development paradigms, defined the phases that are central to most modern software development processes. Boehm [16] proposed the spiral model of development, an evolutionary risk-driven process that incorporates some of the systematic aspects of waterfall development. The Unified Modeling Language (UML), developed by Rumbaugh, Booch and Jacobson, is a notation used in modeling and developing object-oriented systems and has become an industry standard [108]. Many of these existing processes are based on the premise of the increasing cost of change curve which states that system changes will be more costly the later in the development cycle a system is in [12, 45, 53, 108]. This results in a focus on upfront requirements analysis and design. However, these processes are unable to account for continuous requirements and system changes that are often needed throughout the development process.

Agile methodologies have emerged in the last decade as a way to address constantly changing requirements and other key problems including the increasing cost and complexity of software development, communication breakdowns among stakeholders, and missed schedules and budget overruns. Agile methodologies purport to address these software development problems by focusing heavily on quick delivery of working software, incremental releases, team communication, collaboration and the ability to respond to change. Agile methods in one form or another has become increasingly popular in practice. A significant majority of practitioners who have been on agile teams indicate that they produce higher quality software, greater productivity and higher stakeholder satisfaction [1].

2.2.1 The Agile Manifesto

One unique feature of the area of agile design is that agile practitioners are guided by the same set of high-level values codified in The Agile Manifesto [13]. This set of values was developed by a group of leading practitioners to develop common ground among the different emerging agile methodologies

and to make concrete the key principles embodied in all agile methods. The values of the manifesto are reproduced below:

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Part of the reason for developing the manifesto was to make clear what the priorities of agile practitioners were and to partly dispel the notion that agile methods were undisciplined and unstructured. The manifesto states that agile practitioners find value in the items on the right, but they value the items on the left more.

This set of values was further expanded into a general set of principles to further detail the core foundations of agile methodologies. These principles are reproduced below [13]:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.

- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Generally, agile methodologies strive to adhere to these principles and embrace the four key values stated in the manifesto. The principles themselves do not provide specific process guidance. For this, one needs to look at the specific agile methodologies that have been developed.

2.2.2 Agile landscape

A number of different agile methodologies have been developed around these principles that focus on different issues including programming practices, project management, business factors and agile adoption [5, 11, 12, 36, 45, 53, 59, 69, 106, 115, 116]. These include but are not limited to extreme programming (XP), Scrum, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD) and Lean Development (LD) and Agile Modeling (AM). This section focuses specifically on XP and Scrum. These are the mostly widely practiced agile methods and are the primarily methods from which my XSBD approach draws from [69].

2.2.3 Extreme programming

One of the most widely adopted agile processes is extreme programming (XP), developed by Kent Beck with the help of Ward Cunningham and Ron Jeffries [11, 12]. It is a developer-centric approach that is derived from a number of existing software development practices such as cyclical development cycles and feature-driven development and is focused on delivering value to the customer through a flexible development process that reduces the cost of change curve. Extreme programming largely eschews upfront design processes such as those in traditional software engineering processes and instead proposes an evolutionary design process with tightly integrated designer/customer teams. It is built around a set of 5 key values: effective *communication* among project stakeholders, *simplicity* in the design, *continuous* feedback through test-driven development and through evaluations by customers and other team members, *courage* through minimal design and a willingness to address changes, and *respect* for other team members and stakeholders. One can see that these values can be directly tied to the values highlighted in the Agile Manifesto [13]. Extreme programming reduces the cost of change curve through a tightly coupled combination of planning, design, coding and testing practices. Traditionally, this process relies on the entire team—including the customer—to be at the same location, so they are able to work together most effectively.

Planning. Like other agile methods, XP follows an incremental development process that is divided into releases which are in turn divided into iterations [12]. This allows developers to provide demonstrable value to customers in a short amount of time and allows continuous feedback to be obtained. In release planning, project members identify system requirements in the form of stories—which describe specific features that need to be developed. These stories are prioritized by customers. The team then determines which set of stories will be developed in the next release based on customer needs and business value. In each increment, which typically last several weeks, developers break the highest priority stories into development tasks which are estimated and assigned to the developers. At the end of each increment, it is expected that a working piece of software is completed and tested. It is possible for stories to change or for new stories to be identified throughout the process as the software is incrementally developed and reviewed.

Design. At project inception, project stakeholders identify high-level project goals and identify a metaphor—an overall concept of the system that is being built. This will help the team to make consistent design decisions and aid in collaborations with one another. In addition, one key practice of XP is to use the simplest design possible that fulfills a requirement. This is so development work is not wasted on functionality that ends up not being needed in the future. Developers are encouraged to refactor their code regularly so that the resulting designs are as simple and easy to understand as possible.

Coding. XP takes a generalist approach to development where any member of the team is expected to be able to take on any task. One way this is done is through pair programming. In pair programming, two developers work together at a single workstation whenever they are coding. One person is doing the actual coding while the other is in charge of looking at the broader design and coding issues. For example, this person might determine that the code needs to be refactored or may identify a better algorithm for solving a problem. In XP, code is ‘collectively’ owned by the team as a whole. If one programmer sees a change that needs to be made, then he has a responsibility to make it himself. This requires all programmers to be familiar with the code base as a whole.

Testing. A heavy focus on testing makes the development and design practices such as refactoring and collective code ownership possible. XP follows a test-driven development process where programmers first write the test cases for a story, before writing the code itself. These automated test cases are combined into a common test suite that includes all the tests written by all the programmers.

The programmers can then integrate any changes they made into the project code base by running it through the entire test suite. This way, there is always a daily build which includes all the latest changes made to the code and that functions without identified errors.

2.2.4 Scrum

Scrum is an agile development framework, developed by Ken Schwaber and Jeff Sutherland that is becoming increasingly popular among practitioners due to its effectiveness and relative simplicity [114, 115, 116]. Unlike XP, it does not prescribe specific practices describing how to development software. Rather, it consists of a set of practices that focus on the project management side of things. Scrum is based on the idea of software development being maintained through *empirical process control* [115]. Due to the complex and unpredictable nature of software development, the development process itself needs to be continuously monitored and refined to ensure that the team converges on an acceptable end product. This is in contrast to *defined process control* where process can be defined such that the output is at a consistent, desired quality. Empirical process control is achieved through *visibility* of the product and process being used, regular *inspection* of the process and *adaptation* of the process if any problems are identified.

Scrum roles. Roles in Scrum are defined as ‘pigs’ or ‘chickens’ depending on their level of involvement. Pigs are those that are committed to the project and are viewed as part of the Scrum team. Among the pigs are the ScrumMaster, the Team and the Product Owner. The ScrumMaster is not a project manager, but rather a facilitator who helps ensure that the Scrum process is being followed by the team by enforcing rules and keeping them focused. Team members are the actual people that are designing and developing the product. The team is meant to be self-organizing in that they collectively determine what specific processes to use in developing the product. The Product Owner is the customer representative and ensures that the system is meeting their business needs. ‘Chickens’ are not viewed as part of the Scrum team, but have some link to the product. They include customers and vendors who will purchase the product but are not regularly communicating with the Scrum team.

Process framework. Like XP, Scrum is an incremental development process that is divided into ‘sprints’—typically a two-four week period. At the end of each sprint, an implemented piece of the system is completed. Within each iteration, the team determines what features are developed based on the requirements, the available technology and the teams’ capabilities. The set of features that are developed in a particular sprint are taken from the ‘product backlog’, which is a prioritized set of

requirements controlled by the Product Owner. The Team's responsibility is then to develop the highest return on investment (ROI) features selected by the Product Owner. The ScrumMaster ensures that the Scrum team understands the process and follows it as they develop the features.

Each sprint begins with a planning meeting where the team determines what work will be done in the iteration. During the sprint the team holds daily meetings or 'Scrums' where the team reports on project status. In each meeting, which is typically time boxed to 15 minutes, each team member reports on what they did the previous day, what they plan to do today, and if they encountered any problems that are preventing them from getting their work done. In this way, the team can address problems quickly as they occur. The team relies of a variety of 'information radiators' [118] which relate back to the visibility requirement of empirical process control. These might include things such as a task board to show what tasks are completed and which are not, or a burn down chart which displays remaining work in the sprint as a 2D line graph. This chart can be used to adjust work within the sprint as necessary based on the teams' development velocity. Each sprint ends with a review meeting where the team reviews what work was completed and not completed, and demos the completed functionality to stakeholders. In addition, a Sprint Retrospective is held at the end of each sprint where the team reflects on the past sprint and makes process improvements as necessary. This relates back to the *inspection* and *adaptation* requirements of empirical process control.

2.3 Enabling agile usability

One shortcoming of agile development methodologies that became apparent as agile practitioners began developing more interactive and UI-intensive applications is their marginalization of usability issues [4, 56, 102]. This is especially true of agile methodologies such as XP as they were originally developed to focus on satisfying development and business needs rather than on end user needs. Scrum, being more of a project management framework, also did specific guidance on how to integrate usability into an agile team. Agile practitioners and researchers have acknowledged the need to develop systems that meet usability requirements in addition to meeting functional and market requirements and have begun to explore ways of incorporating usability into agile methods [3, 4, 14, 30, 34, 43, 46, 80, 83, 84, 85, 97, 102, 117, 127]. These various agile usability methods can and do leverage some of the key similarities between agile and usability methods but need to address some of the conflicts or divergence points between the two areas to work effectively [90]. The rest of this section highlights some of the key convergence and divergence points between agile and usability and how various methods are used to address them.

2.3.1 Convergence points between agile and usability

Agile methods and usability engineering are built on some of the same principles. One of the key similarities is that both acknowledge that system development is a highly complex and dynamic endeavor that is subject to changing requirements and uncertainties that cannot be known in advance [69, 111]. As a result, both agile and usability methods follow cyclical development cycles, focus on early and continuous testing and are inherently human-centered.

Cyclical development. As stated before, both agile and usability methods follow cyclical development processes. This is a way for the system or UI design to be developed iteratively so the developers can verify that the system functions as specified and so they can make course corrections as new requirements emerge. In usability, this relates to the task-artifact cycle where tasks or requirements determine how the artifact is designed. The artifact, in turn, can affect the task that it was originally designed to support [28]. This similarity makes it easier to integrate usability into agile methods than into more traditional software development methods such as the waterfall development cycle [108]. For example, Sy and Miller have developed an agile usability process where development and UI design operate in parallel with the UI designers working ahead of the developers [84, 127]. With software teams that use the waterfall process, usability requirements and design would take place only at the beginning of the process so by the time developers got around to implementing them, the designs would be out of date due to unforeseen circumstances and changing requirements. Alternately, usability professionals would only be brought on board after the system had been implemented so that any feedback would have limited impact on the project [111].

Continuous testing. Since both agile and usability methods follow cyclical development processes, both rely on testing to verify that the developed system is meeting the project requirements. Agile methods like XP follow a test driven development cycle where code tests are developed before the functionality itself [12]. In addition, acceptance testing is carried out by the customer representative to verify that the system functions as he or she specified. Similarly, usability methods rely on a variety of analytic and empirical testing methods both to evaluate and compare different designs and to verify that the implemented system meets end user needs. In agile usability, usability testing can be viewed as an extension to acceptance testing that contributes to the overall quality of the product.

Human-centered development. Both agile and usability methods are human-centered in that they both rely extensively on communication and coordination between various project stakeholders. Communication is one of the central pillars on which the agile processes are built. Instead of relying

on extensive documentation, teams are expected to communicate and coordinate effort on a daily basis and be able to help each other address problems as they come up. Processes like XP also have an onsite customer who works regularly with the team to define requirements, answer questions and verify that the system functions as he requested [12]. Similarly, usability processes rely on continuous communication and coordination among subject matter experts, usability engineers and end users [111]. Observations of workplaces, usability testing with end users and participatory design techniques ensure that usability engineers understand users, user tasks and the context in which the system will be used. Holtzblatt and Beyer have developed a streamlined version of their contextual design process—which uses ethnographic data collection from end users to drive UI design— called *rapid contextual design* to work more effectively within an agile framework [14].

2.3.2 Divergence points between agile and usability

Given the philosophical similarities between agile and usability methods, many of the difficulties of integrating the two approaches arise due to their different specific practices and the fact that usability methods focus on end user needs rather than customer needs. Some of the divergence points between the two areas are highlighted below along with a description about how different agile usability approaches address them.

Phased vs incremental approaches. Although both agile and usability methods follow cyclical development processes, they differ in what work goes into each of those cycles and how fast those cycles generally go. Agile methods tend to use incremental development cycles where during each iteration some piece of functionality is designed, implemented and tested. This allows customers to give feedback on the system early and often and validate that it does what they want it to do. Usability methods like SBD tend to follow a more ‘layered’ or iterative approach where the requirements are first defined and the system is then completely defined at increasing levels of fidelity [38, 111]. By better understanding the user and the context of use, designers can look at things more broadly and deliver a more cohesive UI design.

This divergence point directly relates to the issues of whether the approach should be *agile-centric* or *usability-centric*. Cooper argues that approaches like XP are too development-centric and that developers do not naturally design code to meet end user needs [37]. Cooper and others argues for a ‘usability-first’ approach where usability professionals first interact with customers to collect data from end users and develop the UI design before working with developers to implement it using a traditional agile approach [34, 90, 100]. Beck counters that such an approach represents ‘Big Up

Front Design' and runs counter to agile practice of continuous development and feedback and would be a waste of resources as developers would have to wait until the UI design was ready [90].

Patton has advocated a more agile-centric approach where existing agile teams learn about and integrate usability practices into their day to day tasks. Patton as well as Meszaros and Aston have reported on projects where developers have used some user-centered techniques as they developed systems using an agile approach [83, 102]. One potential problem with this approach is that effective UI design can be difficult for complex systems with a heavy emphasis on usability. They may require a level of expertise and knowledge that cannot be learned in a short period of time. User interface design and evaluation is not a simple endeavor for systems with a large UI component. Moreover, certain usability tasks are complex and time-consuming and may not be easily handled by developers who also have to implement features.

Agile usability practitioners more commonly take a hybrid approach whether usability and development operate in parallel [97]. Ambler advocates a more integrative agile approach to usability engineering where just enough requirements analysis and modeling is done up-front to begin development [3, 4]. Subsequent user interface modeling, design and evaluation occur continuously in parallel with development. Similarly, Lynn Miller et al. at Alias, suggest a methodology where software development and usability engineering proceed in parallel tracks [85, 127]. The usability team designs the interface for the next iteration which the developers will then work to implement. Usability specialists also validate the usability of what was developed in the previous iteration. These approaches generally acknowledge software engineering and usability engineering are distinct and separate areas of expertise. In this case, communication and careful coordination are vital as agile developers and usability specialists can have differing motivations, thought processes and goals. In addition, these approaches need a well defined way for maintaining a consistent and coherent interaction architecture since there is minimal up-front design [90].

Shared understanding vs distinct roles.

Agile methods lean towards a generalist approach to software development where all of the developers not only have a shared understanding of the design but are equally qualified to work on any part of the system [46]. Benefits of this approach include improved communication between team members and increased flexibility in terms of what works is done by whom [2]. However, usability engineering is a distinct discipline that software developers are typically not trained to do. Usability engineers are

needed for projects where usability is a key quality attribute and the user interface design is non-trivial. However, usability engineers are often not trained as skilled software developers [46].

There are several ways the conflict between the concept of generalization versus the apparent need for specialization with the integration of usability has been addressed. Many agile usability approaches including the parallel development approach used by Sy and Millar among others have distinct developer and usability roles within the team [84, 97, 127]. Although this specialist approach allows each group to leverage their own areas of expertise in developing the system, it requires careful coordination between the different groups to prevent problems such as drift between the UI design and the implementation. In addition, it can be difficult for usability engineers and agile developers to work together given their differing focus areas, backgrounds and concerns [30, 73].

As noted above, a more traditional generalist approach is one where developers are trained in some usability practices so they can effectively take on the roles of both developers and usability engineer [83, 102]. This sort of approach is more suited to systems where usability is less important to the customer and where the user interface is relatively straightforward to design. Ambler advocates a similar approach in that usability engineers can be trained to take on some development tasks in addition to their usability work [3]. Both approaches highlight a potential conflict of interest that can arise between UI design and software development. Given a limited amount of time to complete a feature, the developer is more likely to sacrifice usability to get the code implemented since functioning code is central to agile methods [13, 36].

Maurer et al identified a third possibility in the ‘generalist specialist’ [46]. This is someone with both formal training in user centered design and in software development. This particular person was working in a large organization and acted as a liaison between the usability and development groups within an agile team there. Unfortunately, finding such an individual is likely a rare event given the amount of effort required to adequately learn both disciplines.

Working software vs design documentation. One of the foundations of the Agile Manifesto is that working software is valued over comprehensive documentation [13]. Past software development projects would often get bogged down by large requirements and design specification documents that were difficult to maintain and would quickly become out of date. In agile methods, high quality working software is valued above all else since that is what is being delivered to customers [5, 36]. Agile methods like XP strive to minimize documentation to only what is absolutely necessary through

practices like onsite customers and close collaboration between team members. Usability engineering methods like scenario-based design would appear to work against this principle as they use a variety of different design artifacts to develop the UI interface before any code is written. In SBD, a variety of different types of scenarios are developed to describe current work practices and the system being developed [111]. In addition, a variety of low and high-fidelity prototyping techniques such as sketches, storyboards and click-through mockups, are used to design and evaluate the interface before it is implemented. These types of prototypes are typically used in formative usability evaluations to get early feedback on designs from users and other stakeholders before implementation begins.

Ambler has developed *agile modeling*—a methodology for modeling and documenting software systems—that was developed to be used in conjunction with agile methods such as XP and Scrum [5]. This framework describes how lightweight modeling that is ‘just barely good enough’ combined with things such as active stakeholder participation and test driven development can maximize the utility of documentation within agile teams. He defined an agile usability framework that builds on this idea and is compatible with lightweight usability methods such as sketches and storyboards to quickly prototype designs within an iteration. Such practices are increasingly used in agile usability teams as a quick way to iterate on designs [97]. However more detailed artifacts and practices such as high-fidelity prototyping and user modeling are difficult to do within an incremental development cycle.

Test driven development vs. usability evaluations. Test driven development is one of the most common agile development practices. Agile developers continuously create automated unit tests that define what the software is supposed to do before writing the code itself [12]. This practice allows developers to incrementally develop the system while ensuring that the code base remains robust even as it is evolved and refactored. It also allows developers to identify design flaws sooner, discover problems in the requirements and diagnose and fix problems in the code more quickly. In addition, these test suites can be run automatically on a daily basis. There has been some work tools that can automate the usability testing process. However, such tools depend on developing accurate models of human behavior which in itself is an active research area [55]. Usability testing does not have the benefit of the level of automation that test driven development supports as they require human intervention both in the sense that they rely on feedback from actual users and they often depend on a usability expert to analyze and interpret that feedback. As a result, usability processes tend to take a less nimble approach than agile methods by focusing more on gathering up-front data beforehand and doing early lightweight prototyping iterations before code development begins. For example, the agile concept of refactoring does not have a clear analog in usability approaches since the impact of making

small changes to the user interface can often not be verified until after the next set of usability evaluations are run with users.

Lightweight, or guerilla usability techniques, are commonly used in agile usability teams as a way to get usability feedback within an agile framework [93, 97]. These might include analytic techniques such as cognitive walkthroughs or rapid iterative testing and evaluation—where fixes are made as they are found in a study so that subsequent participants work on a continuously improving system [81]. These techniques allow usability engineers to get feedback from users and provide guidance to the developers quickly which is essential in agile teams. In fact, one potential benefit of this approach is that the system usability can improve more than using a traditional approach since the team is getting feedback from users earlier and more often [127]. However, it is more difficult to run summative in-depth usability evaluations within an agile framework since development moves so quickly and since the UI is developed in a piecemeal fashion [97]. In this way, it can be difficult to get a good overview of the overall design of the user interface and how well it is meeting usability goals.

Customer focus vs end user focus. Both usability and agile approaches are ‘human-centered’ in that they value close collaboration with stakeholders. However, they differ in which stakeholders they focus on most. One of the core concepts of agile development is close collaboration and continuous with customers. In XP, there is an on-site customer who joins the team and works with them throughout the development project to define requirements, do acceptance testing and answer questions as they arise [12]. In agile teams, the ultimate goal is to efficiently develop a high quality product that meets the user’s needs. Usability engineering methods like SBD are user-centered, meaning that the ultimate goal is to develop a high quality system that meets the end users’ needs. Usability engineers use a variety of user-focused techniques such as onsite observations, interviews, participatory design and user testing to understand users and ensure that the system meets their needs [111]. For many development projects, the customer and client company will not be the ultimate end users of the developed system. This can result in conflict between the usability engineer and the agile developers because of their focus on different stakeholders.

The differing focus points between agile and usability can lead to communication and collaboration problems within agile usability teams. Agile developers and usability engineers can come into conflict when their goals do not align. For example, although simplicity in the design is a characteristic that is valued by both agile and usability practitioners [13, 35], simplicity in the user interface often does not align with simplicity in the implementation. In addition, usability and agile professionals can have

problems understanding or accepting each others' practices and worldviews. For example, usability engineers need to understand that business and technical factors can impact the importance of usability as a quality factor in the system. In addition, agile practitioners need to understand that working customers does not guarantee that the resulting system will be usable for end users [56]. These sorts of social and cultural factors can have a large impact on team cohesiveness. Although they have been studied within the context of agile teams, more work is needed to study how they affect agile usability teams in particular [46, 74, 75, 134].

2.4 Chapter summary

This chapter provided an overview of foundational work in the fields of usability and agile methods. It provides an overview of usability engineering, agile methods and key challenges that need to be addressed by an integrated agile usability approach. This dissertation work has been focused on integrating scenario-based usability engineering into an agile development framework by drawing on theoretical work from leading HCI and usability practitioners and agile practitioners. It will leverage key similarities between agile and usability methods and address the divergence points identified here through a combination of shared design representations, project management framework, development practices and coordination of usability and development efforts. It will focus on enabling incremental development of the UI while also maintaining a high-level view of the interface and how well it meets high-level goals. In addition, by bringing active research in HCI and design methods into practice, this work addresses the common criticism that HCI and usability research does not apply to practical situations—resulting in an apparent ‘theory-practice gap’—by relying on shared design targets, knowledge capture, and practical development concerns that cut across design projects [23, 26, 27, 60, 70, 101, 110, 135]. The next chapters will detail my work in scenario-based design, the central design record, and how they were combined with XP to form the XSBD process.

3 Background work

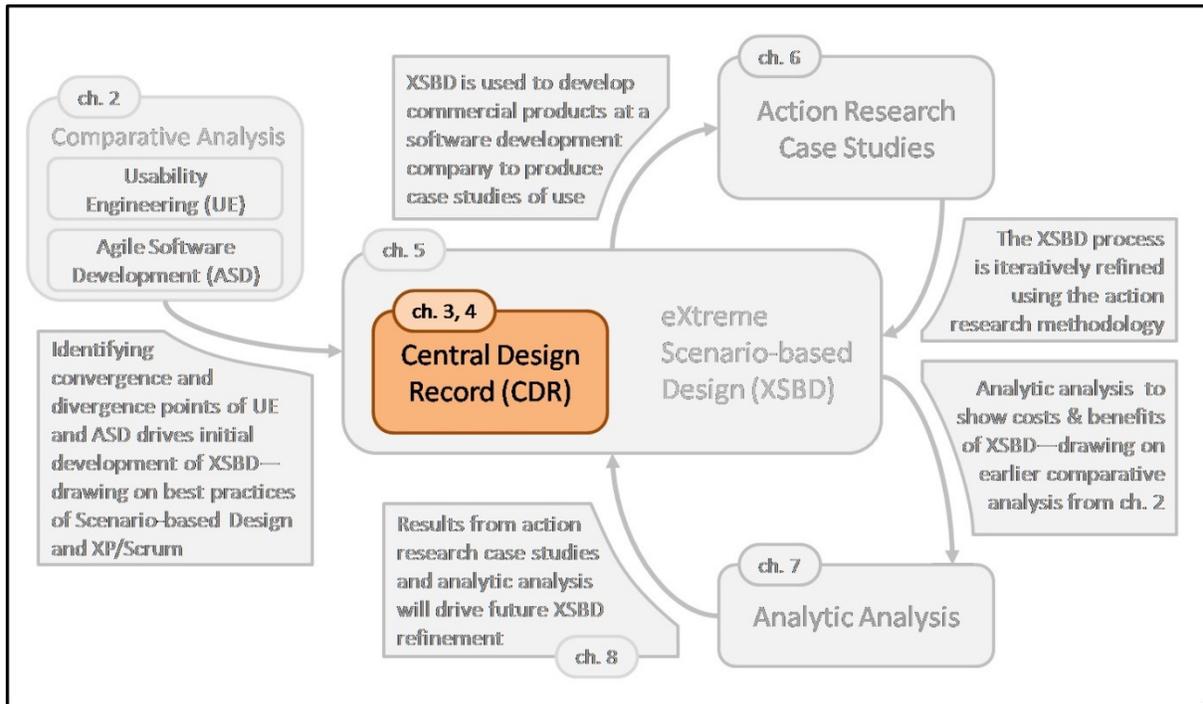


Figure 4. Overview of solution approach. This chapter summarizes my initial work on scenario-based design and design representations which would later inform my work on the CDR and agile usability.

This chapter summarizes my initial work in scenario-based design, design representations and knowledge reuse (See Figure 4) [62, 63, 79, 89]. It describes my work on the LINK-UP system, an online scenario-based toolset that supports SBD and knowledge reuse. This work led to my initial work on design representations—based on Norman’s work on the *system image* [99] and Carroll and Rosson’s work in scenario-based design [24, 111]. This work and associated research studies, involving students working on semester-long development projects, provided the foundation from which my later work on the CDR and integrating usability and software development methodologies is based.

3.1 Leveraging a design representation in LINK-UP

The goal of this initial work was to design and develop a tool to enable the learning of usability engineering and HCI concepts while contributing to a growing and evolving development environment called LINK-UP. It focused on teaching students and novices about analytic evaluations, a popular usability evaluation method that can be conducted rapidly and inexpensively. These evaluations were centered on the use of claims as a point of discussion of design features among evaluators and designers. This work further discusses how this system supports design knowledge reuse through

claims. This section is adapted from my E-Learn 2004 conference paper: *From Chaos to Cooperation: Teaching Analytic Evaluation with LINK-UP* [63].

3.1.1 Problem overview and research goal

One of the concerns of human-computer interaction (HCI) is the development of interfaces that allow people to effectively use computers to meet their goals [22]. This multidisciplinary field of study is difficult to teach in higher education settings and may appear to students as a seemingly random collection of topics, without the structure important to facilitate learning. Our efforts seek to design and develop a tool that will enable students to learn basic concepts of HCI while contributing to a growing and evolving effort. Key in our system are analytic evaluation methods for software interfaces, which enable students or design professionals to cost-effectively conduct a usability test in minimal time with few participants [96]. Like students of other design disciplines, interface designers encounter valuable learning opportunities when they have access to many examples of existing design products and multidisciplinary perspectives. Therefore, information from the evaluation is stored for reuse in other phases of development, and by future system developers. With this high-level vision, our efforts target development and initial testing of such an analytic evaluation tool within an emerging web-based, integrated design environment—LINK-UP.

3.1.2 Background

Notification systems design with LINK-UP

To develop and validate our prototype design environment and analytic testing tool, we have constrained focus to a specific challenge within the field of HCI—developing notification systems. As a specific type of computer interface, *notification systems* typically are used for very brief information interactions, unlike most interfaces that are used for extended periods. Users value notification systems for reacting to and comprehending time-sensitive information, without introducing unwanted interruption to ongoing tasks [76, 77]. Examples include news tickers, alerts or pop-up windows, automobile displays, system or network monitors, and context-sensitive help agents. We have found that notification system design problems can make very rich assignments for HCI students—requiring thoughtful consideration of multimodal design tradeoffs and information visualization techniques to effectively present interface state transitions to users [32].

LINK-UP (Leveraging Integrated Notification Knowledge with Usability Parameters), is an integrated design environment for notification systems that allows designers to proceed through development activities while accessing and creating reusable design knowledge with claims as the basic reusable

knowledge unit—an idea first proposed by Carroll in his Task-Artifact Theory [28]. Complementary to a scenario-based design approach, the LINK-UP system provides interactive modules and tools for activities that include requirements analysis (brainstorming) with task-models, participatory negotiation with end-user stakeholders and analytic and empirical evaluation [31]. Through the progressive use of LINK-UP, developers create and reflect upon interface artifacts defined at ever-increasing fidelity. LINK-UP helps developers focus HCI concerns on the critical parameters of notification system design—controlling appropriate levels of user interruption, reaction, and comprehension, or *IRC parameters* [77]—with its integrated access to detailed notification design tradeoffs (or *claims*) established through hundreds of previous design efforts, empirical testing results, and concise summaries of psychology and human factors literature. As developers use LINK-UP, these basic research results are applied and put to the test with new design efforts, continuously advancing the state-of-the-art in notification research. To widen the scope of reuse, the knowledge contained within claims and their associated artifacts must be classified and generalized; fortunately, a schema and method for classifying claims is introduced by Sutcliffe and Carroll [126]. LINK-UP is intended to provide a framework for design knowledge reuse, coupling a claims library with a suite of design-support tools [31].

Encouraged by initial acceptance of our broad approach to computer-aided design of software interfaces, our recent efforts focus on development and validation of specific modules within LINK-UP. Arguably the most critical module within LINK-UP, the work reported here relates to the *Analytic Module*.

Analytic Evaluation in LINK-UP

Generally, an analytic evaluation is a process in which evaluators (often experts) identify usability problems in a partially implemented interface by ‘simulating’ the way they think users will perform certain tasks. One of the fundamental ideas in HCI is that usability evaluations should start early in the design process, optimally in the stages of early prototyping. When critical design flaws are detected earlier, the chance increases that they can and will be corrected without requiring costly evaluation with actual users (i.e. empirical evaluation) or major code changes. The analytic evaluation module will ultimately support several different types of evaluations, such as the use of heuristics [96].

However, in this prototype, we employ a critical parameter comparison technique that contrasts the *design model* (expressing the designer’s interpretation of user requirements) with the *user’s model* (characterizing the actual user’s experience)—concepts originally introduced by Norman [99]. The

Analytic Module within LINK-UP provides a tool that facilitates execution of an analytic evaluation method, records evaluation results, and provides an estimate of what the user's model IRC parameters might be if the system were fully developed and tested with users (the *analytic model*). That is, to support the goal of early evaluation and introduce students to this essential practice, designers are able to get a preliminary sense of the levels of interruption, reaction, and comprehension that their system design will cause within users. To obtain the analytic model, designers prepare an overview of the key interface interactions (or a *system image*, which may include a system prototype description, interaction scenario, and prototype artifacts). Expert evaluators use this overview to walkthrough and analyze the interface with a series of questions established as an IRC analytic calculation method [33]. During the evaluation, experts also consider the designer's claim statements about psychological effects that the interface will cause in users. Claims can relate to problem spaces (e.g. describing situational characteristics and requirements) and be characterized by a design model IRC, or claims can express proposed design solutions. The module assists designers with matching problem claims with design claims—a process that will assist designers later in pinpointing the underlying causes, if the design model and analytic model IRC values do not match after the expert evaluation.

3.1.3 Analytic module development

We began brainstorming about the design of the analytic module by considering all of the broad activities that users would need to accomplish as they proceeded through an analytic evaluation. Two important realizations emerged: first, the system would be used by two distinct types of users; second, the module would need to support several activities beyond just the evaluation execution. Each of these ideas is elaborated on below. Most of our early design work was conducted with rapid prototype development software that allowed quick storyboard sketching and linking of steps within the module (See Figure 5). In the early process, we were unconcerned with the specifics of information design—we focused solely on ensuring that all essential processes and steps were included. We received frequent feedback on several iterations of our rapid prototypes from members of our larger research group. As our ideas solidified, we emerged with a coherent task analysis and screen-flow for the analytic module.

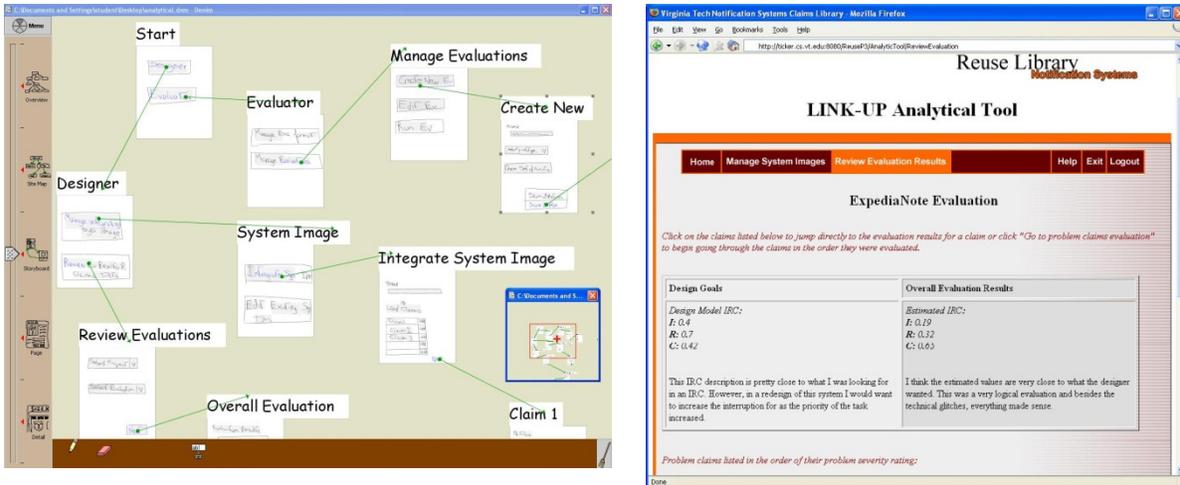


Figure 5. Left: early storyboard of the analytic module; Right: Screen from the functional prototype used in pilot testing.

There are two types of users that will use the Analytic Module, designers and evaluators. Designers are the individuals who will develop the system to be evaluated using scenario-based design and LINK-UP's claims library. Evaluators are the individuals who will be assessing the system's usability. Ideally, they would be experts in HCI and be familiar with scenario-based design claims analysis [111]. According to our needs analysis, we determined that the analytic module should be divided into three sub-modules: *Manage System Image*, *Evaluate System Image* and *Review Evaluation Results*. The three sub-modules correspond to the basic sequence of tasks (See Figure 6). The designer's and evaluator's tasks will be described respectively below.

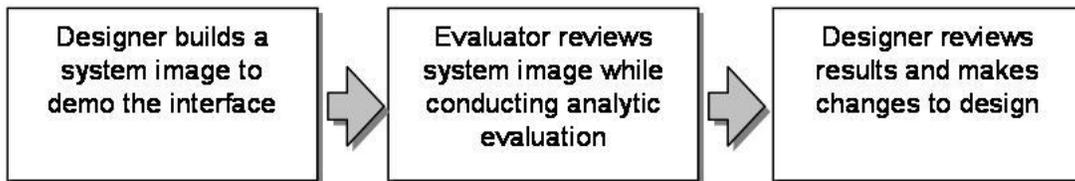


Figure 6. Basic sequence of tasks of the analytic module of LINK-UP.

First in the task sequence, supported by the *Manage System Image* module, designers will either add a new system image into the module or edit an existing system image. To create a new system image, the designer will need to select an ongoing design project. Each design project will be associated with a list of problem claims developed during the requirements gathering and participatory negotiation phases of the LINK-UP system. The designer then links necessary design claims to each problem claim. Each design claim should solve part of or the whole problem addressed in the problem claims. Design claims can either be newly developed by the designer or be reused, drawn from the claims

library. Artifact representations are also uploaded to describe the system image. An artifact representation shows some aspect of the system that is being designed. They can be pictures, screenshots or movies of the system interface. Before this process, the designer would have already established a design model IRC, which represents the intended IRC value for the system being designed. Once this process is complete, the system image will be ready to be evaluated. Designers can also edit existing system images after creating them in response to evolving requirements or to account for system redesign decisions.

Second, through the *Evaluate System Image* module, HCI experts can analytically evaluate system images. The analytic module will step the evaluator through the evaluation method and record the results of the assessment. During the process, the evaluator would first become familiar with a summary of the system image to develop an overall understanding of the notification system design. This is accomplished by browsing through the system description, the interaction scenarios, the design model IRC values, and the prototype artifacts. Then the evaluator steps through each problem-design claim pair and rates how well each design claim addresses the associated problem claim on a four-point scale (*major flaw, flaw, minor flaw, or no problem*). This rating is based on how well the upsides (positive psychological effects) expressed by the design claim either:

- Support the upsides of the linked problem claim
- Mitigate the downsides of the linked problem claim or other design claims

Evaluators provide comments for each claim pair to explain their ratings. At the end of the evaluation, the evaluator runs the IRC equation algorithm to get the estimated IRC value for the whole system [33].

As the final step in the task sequence, in the *Review Evaluation Results* module, designers will be able to review the results of evaluations that experts have run on their system images. At the start of this interaction session, the designer will see overall evaluation comments and will be able to compare the design model IRC and estimated IRC for their prototype. This gives the designer a sense of how well the system might fulfill its most critical user goals, if fully developed. They can then view the ratings and comments for each claim pair to elaborate on specific, causal design features. The designer can use the ratings and comments to improve their designs. They can also revise the problem-design claim linkages, or even modify their original design claims. Since the claims and their evaluation results are archived in the claims library and accessible beyond the single project, future developers will be able to benefit from this stored knowledge.

3.1.4 Pilot testing evaluation methods

We conducted a three stage formative empirical usability evaluation of the analytic module—a process in which potential users actually used the prototype to accomplish real tasks. This evaluation was conducted after the development of the module prototype to get feedback from actual designers and study how it is used in a realistic situation [111]. The evaluation had two goals, to establish whether:

- Designers can use the module to help identify and resolve design problems
- Through use of the module, could designers learn more about claims-based usability engineering

The first goal arises because an overall objective of the LINK-UP system is to support the usability engineering process for notification systems and enhance HCI education activities [31]. The second goal arises because many initial users, especially those in an academic setting, are unlikely to be experienced with claims and scenario based design. The analytic module addresses the user's tendency to learn about a new tool through active and informal, exploratory use [29].

In our pilot testing, we were assisted by seven participants who were all undergraduate students in a semester-long undergraduate research seminar, focusing on design and evaluation of notification systems. The students had spent recent weeks individually designing a notification display that helps users monitor and react to constantly changing airplane ticket prices. At the start of our pilot testing, each student had a semi-functional prototype notification system to support this typical notification task. The evaluation was run over a period of three weeks, with one hour-long session held each week. To prepare for the evaluation, participants added to the claims library their problem and design claims. In *Session 1*, the participants created the system image for their prototypes using the Analytic Module. The participants had the option of linking their own (new) design claims or reusing existing design claims to their problem claims. After the system images were created, each participant completed a questionnaire containing multiple-choice statements (rated on an eleven-point Likert scale) and general questions that required written answers. This questionnaire assessed the designers' opinions about how well the system image represented the prototype design. In *Session 2*, each participant evaluated the system image of a prototype developed by another participant in the last session. This was also followed by a questionnaire that gathered feedback on how effectively the participants were able to evaluate the prototype using the module. In *Session 3*, the participants reviewed the evaluation results for their system images and completed a form to express comments and reactions to all information resulting from the evaluation (quality of evaluator feedback and utility of the general approach). This session was followed by a questionnaire that focused on whether the

evaluation gave them useful design feedback and whether evaluators identified any problems with the claims. A detailed overview discussion of the study results is available here [63].

3.1.5 Summary of study findings

Overall, we can report that the designers and evaluators were able to understand and use the prototype of the Analytic Module as it was envisioned. The pilot testing showed that distributed project work is feasible with the support of at least this module of LINK-UP. We were also pleased to observe a few cases of design knowledge reuse, and we are confident that more cases will be apparent if the tools are more fully integrated in design practice and as the claims library content develops further. To summarize the major findings:

- A prototype representation (system image), expressed in terms of problem and design claims provides enough information for knowledgeable HCI persons to quickly evaluate prototype designs.
- The Analytic Module can help designers identify and resolve problems with their prototype designs in terms of design claims and how they address both the upsides and downsides of problem claims.
- The Analytic Module can help students learn about scenario-based design with claims analysis by giving targeted feedback through evaluations that can uncover misconceptions about claims and relationships between problem and design claims.

The results of our online evaluation tool demonstrates its potential for teaching HCI design concepts such as claims analysis and knowledge reuse to students in a distributed and collaborative environment. However, based on participant feedback, the Analytic Module should separate the process of rating the quality of claims from the process of rating the quality of the design they represent. Furthermore, supporting multiple evaluations for a single system image would strengthen the believability of the ratings.

3.2 Bringing design representations to the forefront

Our continuing work in developing LINK-UP, an integrated design and reuse environment, addresses the need for proven, dependable engineering processes for interface development [62]. It suggests that a better understanding of the system image is key to the successful evaluation of design prototypes, and an aide in applying knowledge from the repository. This section describes our work to enhance LINK-UP by developing and augmenting the system image to make it the central communication point between different stages of design and between different stakeholders. We report on a study of

the modified task flow that demonstrated the value of the system image within a broader design context. Overall, our findings indicate that the effective creation and use of knowledge repositories by novice HCI designers hinges on successful application of existing HCI design concepts within a practical integrated design environment. This section is adapted from my ACMSE 2005 conference paper: *Image is Everything: Advancing HCI Knowledge and Interface Design Using the System Image* [62].

3.2.1 Problem overview and research goal

This work is a continuation of our effort to develop a structured, methodical way to both apply and further develop HCI knowledge through LINK-UP, an integrated design environment that supports a scenario-based usability engineering process centered on the use and development of a knowledge repository. The overall goal of LINK-UP is to validate the use of a reusable knowledge repository as a basis for developing a more principled approach to design that can build on past efforts. Key in creating this is adequately capturing the vision of the designer and relating it to the experience of the user to enable reflection and iteration on the design. This will in turn support the use and creation of reusable knowledge.

In our preliminary work (described in 3.1), we integrated Norman's ideas into the analytic module to allow HCI experts to evaluate nonfunctional prototypes using a digital representation of a system image [99]. An initial study confirmed that the system image supported effective analytic evaluation of initial design prototypes, but also suggested that it was instrumental in the application of information from the knowledge repository. We worked to redesign the initial architecture to further develop our own interpretation of Norman's system image, as well as to apply other concepts derived from HCI so that LINK-UP better supports interface design and the generation of reusable design knowledge. We conducted a study of the new task flow with a group of novice HCI designers that demonstrated the value of the system image as a central part of the LINK-UP system and its effectiveness at supporting interactive system design. However, participants encountered problems during the design process caused by differing understandings of key HCI concepts that were integrated in LINK-UP.

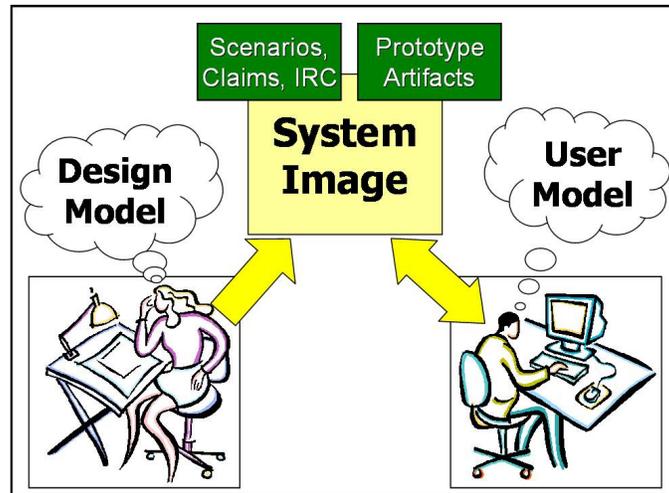


Figure 7. Norman’s conceptualization of the system image. The designer captures system intentions in a prototype, along with supporting documentation, enabling the study of user interaction with the system.

3.2.2 Background

Norman established the idea that governing the usage experience is the consistency of two conceptual models—the *design model* held by the designer and the *user’s model* based on the user’s understanding of the system [99]. Each model can be analyzed as stages of action, which describe the cyclical evaluation and execution of tasks across the *Gulf of Execution* and the *Gulf of Evaluation*. To facilitate a user’s evaluation and execution of tasks, designers must develop conceptual models as they would develop the scaffolding of a bridge. Several factors contribute to each of these conceptual models. The design model should be inspired by a requirements analysis, including consideration of a user’s background, situational context, and task-oriented goals. This model expresses the designer’s understanding of user needs and is a representation of the intended functionality for the system. The user’s model is formed by the user’s understanding of the *system image*, the physical system and its documentation (See Figure 7).

The key idea we continue with is that Norman’s view of the role of an interface designer is to develop the system image so that the user’s model and design model are compatible. Carroll’s work focuses on a scientific approach to research and interface development that drives interface innovation and HCI research through an approach like his scenario-based design process. Through the development and sharing of narrative descriptions of users solving problems with designed systems, designers are able to create the scaffolding across Norman’s Gulfs—and develop systems with design-user’s model compatibility. The convergence of Norman’s ideas with Carroll’s, provides the theoretical basis for this effort. The iterative process of gauging critical parameters, embodied in design artifacts and

expressed with claims, should guide an evaluation of the system image and provide indices for reusable design knowledge.

3.2.3 Enhancing LINK-UP

In the previous study described in Section 3.1, the participants failed to effectively leverage critical parameters as a way to guide the analytic evaluation process and the subsequent redesign of their systems. This was because students could not see the connection between the system IRC values, which describe the IRC values for the entire system, and how they could be applied to design decisions involving individual features. We began redesigning LINK-UP to address this shortcoming and further investigate the incorporation of existing HCI concepts so that it better supports design processes and the creation of reusable design knowledge.

To this end, we incorporated Carroll's interpretation of the stages of action [111] within the LINK-UP development process. As a result, designers would express requirements for each stage in terms of the IRC critical parameters. This was done to provide additional guidance to designers in developing and applying claims and to better support the use of critical parameters within the design process.

Within LINK-UP, the system image consisted of the scenarios, claims and IRC values developed for the system in addition to associated screenshots and drawings depicting the system. The task-flow was modified such that the system image would act as the central component of the LINK-UP architecture through which the different modules would interact. For example, the result of an evaluation from the empirical evaluation module might suggest some required design change. The designer can then run a participatory negotiation session with end users using the system image as the communication point between them. Changes that are made to the prototype design can be reflected in the system image, which can then be used by any of the other modules in the next design iteration. LINK-UP therefore better supports the rapid, iterative development cycles common in software engineering and which are encouraged in scenario-based design [111].

3.2.4 User study description

We conducted a study to validate that the central role of the system image, combined with the addition of IRC values within a stage of action framework, supported the design of notification systems. We also expected to encourage the development of claims so as to contribute to a knowledge repository. In addition, the study was intended to highlight difficulties novice or inexperienced users may have in using the LINK-UP system to design an interface.

Participants for this study were nine computer science graduate students in an upper-level graduate HCI course. Participants had little to moderate amount of HCI knowledge and had little to no experience applying that knowledge to interface development. The study was run as a series of two project-based homework assignments. For the first assignment, participants were asked to define aspects of a problem situation to motivate the design of a notification system through the development of problem scenarios, claims and high-level goals in the form of critical parameters. This represented the *design model* for their prototype.

For the second assignment, participants developed a scenario describing the use of their prototype, and design claims that were linked to each of the problem claims they previously developed. The nonfunctional prototype, which consisted of a series of screenshots or sketches, combined with the scenarios and claims made up the *system image* for the design. Third, participants exchanged system images and evaluated each other's prototypes. The written usability reviews formed the *user model* for each design. Last, participants were asked to compare the user and design models for their prototypes, and look for possible improvements to their design based on the results. They then completed a questionnaire that focused on how they used the system image how it helped in evaluating the system. A detailed discussion of the study results is available here [62].

3.2.5 Summary of study findings

The study demonstrated the value of making the system image a central part of the LINK-UP system and making it the communication point between the designers and evaluators. The scenarios and claims, which were sorted by stage of action, allowed evaluators to give targeted feedback to designers that helped them identify areas for improvement in their prototypes. The system image supports a similar kind of interaction in other stages of the design process such as in participatory negotiations between designers and clients. This is because it encourages discussion of the most important features of a design, using a common language that is understood between all parties. This common language, or *common ground*, is critical for efficient communication between parties [86].

In addition, the majority of the participants agreed that the system image would be useful later on in the design process even if a functional prototype were available, because the system image would always present only the most important aspects of the functional prototype in the form of claims and critical parameters. In addition, the system image supports efficient changes to both the design and to the requirements of the system since it ties the design and user model together within the stages of

action. This is important in practical, iterative design processes where changes may affect any aspect of a system's design.

The key problems encountered during the evaluation process were caused by differing understandings of the critical parameters and their application to the stages of action. Scenarios and claims are largely written in the form of narrative text, which allows for explanations and elaborations. Indeed, the claim itself elaborates on design decisions presented in the scenarios and in the prototypes. On the other hand, critical parameters are quantitative in nature and their effectiveness is dependent on a universal understanding of what they mean and how they should be used. This common understanding was not established between the participants in this study. As a result, the differing interpretations rendered the IRC values within the stages of action ineffective. Instead, participants relied primarily on the prototype screenshots, scenarios, claims and on communications between the designer and evaluator.

Overall, we can report that the system image, acting as a central component of design, supported the design process presented in LINK-UP. To summarize the major findings:

- A design process centered on the system image, which represents the current design prototype in terms of scenarios, claims and critical parameters, allows designers to iteratively improve designs by having evaluators give targeted feedback.
- The system image supports communication between different stakeholders (designers and evaluators) by providing common ground based on the critical aspects of design.
- The ability of novice HCI designers to use critical parameters and stages of action within a design and evaluation environment depends on a process or tool that supports the development of a common understanding of those topics within a realistic situation of use.

3.3 Other preliminary work

Other preliminary work has focused primarily on the development of LINK-UP and how it supports reuse. "Use and Reuse in Information and Interaction Design" describes the use of claims as a knowledge capture mechanism [79]. It focuses on how claims are created, how they interrelate with each other and how they can be reused across different design projects. This work provides justification for the use of claims within LINK-UP. "From Personas to Design: Creating a Collaborative Multi-disciplinary Design environment", describes how personas were used in

developing the next iteration of LINK-UP [89]. Personas are descriptive models of target user groups derived from collected data from interviews and observations. We developed four personas to target the next version of LINK-UP to: the notification system designer, the HCI researcher, the psychologist, and the software project manager. This reflects our growing need and interest in developing a way to address the needs of both researchers and practitioners.

3.4 Chapter summary

This chapter summarized my preliminary work in scenario-based design, design representations and knowledge reuse that was completed through my work in LINK-UP. The primary purpose of developing LINK-UP was to develop a framework to support design knowledge capture and reuse by incorporating it within the development process itself. My own work focused more on aspects of the development process itself, particularly analytic usability evaluations and the use of the *system image*, a design representation based on work from Don Norman, John M. Carroll and William Newman [24, 91, 92, 99].

This work highlighted how a design process centered on the system image allows designers to iteratively improve designs through targeted feedback based on specific claims. It also demonstrated how the use of a common design representation such as the system image can help designers and evaluators by providing common ground through which they can communicate more effectively. These results, along with subsequent requirements gathering efforts [89] were used to drive the next version of LINK-UP.

Overall, this work served as a foundation for my subsequent work in combining usability engineering and agile software development processes using the *central design record*, a design representation that evolved directly from the *system image* and that was incorporated into the next version of LINK-UP. This subsequent work is a departure from the initial focus on reuse of design knowledge—but is a direct outgrowth of and remains complementary to that work.

4 Developing XSBD around the CDR

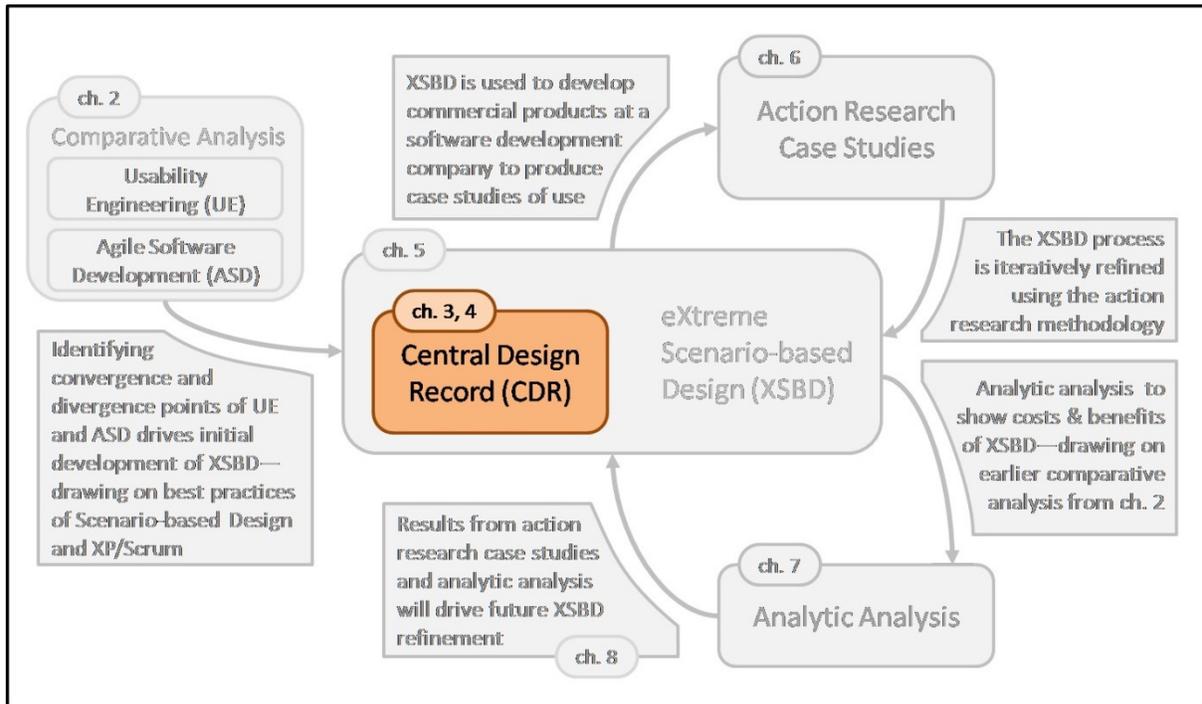


Figure 8. Overview of solution approach. This chapter summarizes how I developed the CDR and used it as a core feature of the XSBD agile usability approach.

This chapter summarizes how the shared design representation known as the central design record (CDR) was developed and integrated into the XSBD agile usability process (See Figure 8) [61][64][67]. The CDR was initially added to the LINK-UP system as the main design representation of the scenario-based development process. It describes how the CDR was used by student developers in designing systems and the problems and issues that became apparent. It then describes how the CDR was modified to act as a central design representation that would be shared by different stakeholders in an agile software development project—with a focus on usability engineers and software developers.

4.1 Defining the Central Design Record in LINK-UP

This work is the culmination of my early efforts at developing a usability engineering process and tool that is based on the scenario-based development process [67]. It develops and defines the main design representation used in the approach—the central design record (CDR). The CDR is a design representation meant to prevent breakdowns occurring between design and evaluation phases—both within the development team and during design knowledge reuse processes. The CDR is intended to make designs coherent and understandable, thus supporting a principled, guided development process

critical for student developers. This work presents the first extensive usage evaluation of LINK-UP and the CDR. This section is adapted from my IJITSE journal paper: *Understanding Usability: Investigating an Integrated Design Environment and Management System* [67].

4.1.1 Problem overview and research goal

While the previous chapter reported on the design of LINK-UP [31, 104], this chapter introduces the central design record (CDR), a design representation that makes explicit where and how handoffs—such as the one between evaluation results and the subsequent redesign of a system—occur in the development process and highlights design decisions that need reconsideration during the next development iteration. It is a structured organization of design knowledge (derived from the knowledge library or newly created) that describes critical features of an interface, what problems the features address, and how evaluation and collaborative design information relates to and affects those features. In this way, it makes explicit the relationship between design goals and the developed interface. For instance, the CDR allows developers to determine links between identified problems and how an interface design addresses them, or to see relationships between evaluation data and different aspects of the interface. Student designers will be able to see how different design techniques affect designs and why those techniques might be used.

This section summarizes results from three case studies that illustrate how novice designers used LINK-UP to engineer interfaces [15, 57, 87]. Building on past experiences with knowledge repositories and principled process-oriented development, LINK-UP serves as a culmination and realization of the prior work of McCrickard and Chewar [31, 33, 76, 77]. The results suggest that a design knowledge integrated development environment (IDE) that incorporates the processes and principles of the central design record, can help novice developers apply a design methodology to develop interfaces in a guided, methodical fashion and avoid process breakdowns. Furthermore, there is evidence that, through the use of the CDR, iterative application and verification of reusable design knowledge—important in the education of novice interface developers—is made practical.

4.1.2 Motivation and related work

Various methodologies and techniques within HCI have identified problems and issues that need to be addressed in a usability engineering process. In this section, some of the foundational ideas that motivate the development activities supported by LINK-UP and the central design record are presented.

The creation of a detailed design representation in SBD is imperative to the design process, allowing explicit analysis of the new system in its anticipated context of use. *The designer's goal is to complete this design representation with as much detail as possible.* With a well-defined design representation, the reuse of design knowledge can prove to be immensely helpful as they are more likely to fit into the structure of the design representation and contribute to the overall design. Being able to effectively store and reuse design knowledge is an active area of study for design domains [47, 71, 125, 126]. For example, the software engineering community has long advocated reuse of both code and general code architecture solutions through patterns. Within HCI, Sutcliffe and Carroll worked on a framework for documenting and organizing claims in a knowledge repository, although as of yet they have not developed the actual library or tools to support claims reuse processes [126].

Another important consideration in usability engineering is to support communication among the different groups of stakeholders that may be involved in a development process. Given the interdisciplinary nature of usability engineering and HCI, people from different backgrounds may not have an easy way to discuss and reflect on designs. Borchers advocated the use of formally defined patterns to support communication among stakeholders [18]. In a similar vein, Borchers and Erickson have both advocated the use of patterns and pattern languages as a way to support cross-disciplinary discourse [18, 42]. Sutcliffe pointed to structured claims as a way to delivering HCI research knowledge to practitioners, giving them the ability to communicate through claims [125].

4.1.3 LINK-UP

Based on these prominent ideas and implications emerging from HCI research, requirements for LINK-UP were derived. Performing a decomposition of the overall goal of providing useful support for iterative application and verification of reusable design knowledge, the following focus points with respect to the development of design knowledge IDEs are derived:

1. IDEs must support design goal formation and facilitate continuous estimation of design progress through comparison between design goals and resulting design artifacts.
2. To guide specific, incremental design improvements, the system should help developers craft a design representation that is sufficiently detailed to focus evaluation activities.
3. IDEs should facilitate communication efforts among stakeholders around the design representation and its resulting development and evaluation.
4. An interface development support system must be flexible enough to support design and evaluation activities.

With the four focus points in mind, the goal was to create a system that supports the design of notification systems through the use of critical parameters and Carroll’s notion of claims from scenario-based development. LINK-UP consists of a design knowledge repository and modules in which design activities are carried out. The repository, or the *claims library* [104], contains claims related to the notification system domain. Designers can search for reusable claims applicable in their own designs by using various searching or browsing features [126]. This basis, a structured collection of claims, supports knowledge sharing among designer communities interested in the domain. The *central design record*—developed and maintained using web-based design modules—allow designers to organize and integrate knowledge from the claims library as they develop their systems [62, 63]. The central design record is based on Norman’s theory of action, specifically his concept of a system image and the need to match designer and user’s understanding of a design (Figure 7). The CDR makes connections between the designer and user models explicit, highlighting areas for improvement and gaps in the design that can be addressed in subsequent iterations using the claims library as a primary source of knowledge. Figure 9 shows how the CDR and claims library are used to iteratively develop interfaces.

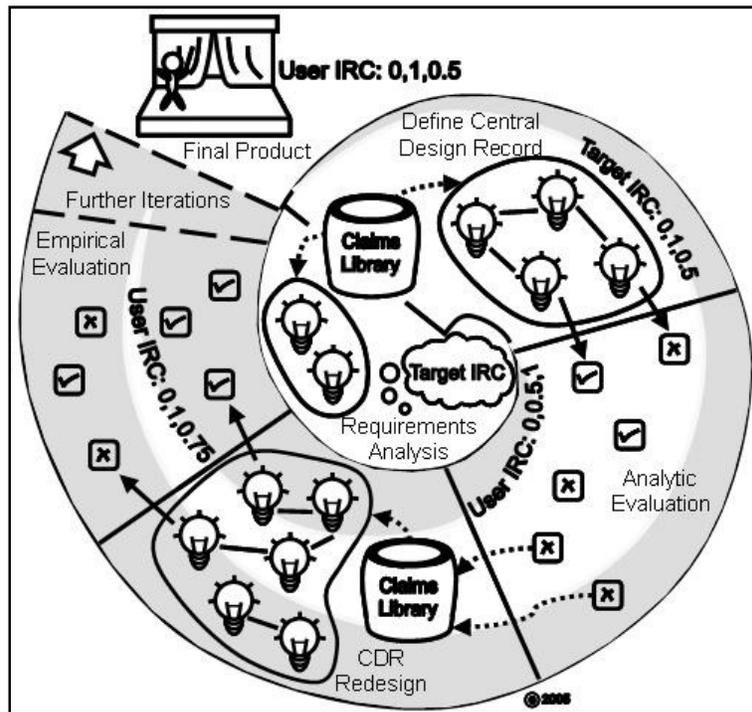


Figure 9. LINK-UP's iterative design process and CDR development. Designers start at the center identifying requirements and a target IRC goal. Design iterations through the CDR include design claims which are tested through evaluations leading to convergence of the design and user models.

4.1.4 The claims library

The basic structure of a claim in the library consists of a feature and a list of upside and downside tradeoffs. This structure is extended with additional information in LINK-UP. Each upside and downside is supported by rationale either summarizing results of an observational study performed by the designer who created the claim or providing references to published research supporting the particular tradeoff. An attached scenario provides context for the claim by describing a task in which the claim can come into use. As a whole, the claim is also assigned an IRC value, the specific critical parameters developed for notification systems, allowing designers to discuss how applicable the claim may be to the overall design goals of their system. Such assignments are critical to integrating the concept of critical parameters into design, providing a base upon which claims can be evaluated. The complete structure of a claim in the library allows designers to create sufficiently detailed design representations through collections of claims.

The claims library forms the core repository of design knowledge available to LINK-UP users. Design processes to guide developers and support their use and reuse of claims are encapsulated in two different modules: the requirements analysis module and the central design record module [62]. Guided by these modules, developers construct and manage the central design records for the interfaces they develop. These consist of an organized set of scenarios describing different usage situations, claims related to specific features in the interfaces, and design goals defined in terms of IRC parameters. The CDR acts as a living representation of the design and is used to guide all stages of design including collaborating meetings, evaluations and design improvements.

4.1.5 Requirements analysis module

The requirements analysis module serves as an introductory module for designers, and is where they first determine design goals and establish the problems that must be solved. By design, the module is organized into a series of detailed, structured steps (See Figure 10). This module introduces many of the important concepts and processes within LINK-UP as novice designers make a first pass at defining system requirements before beginning development work in the central design record module. Subsequent changes to requirements or design goals are done in the more open ended CDR module. Designers are guided through the process of defining a problem scenario and problem claims.

The problem scenario gives insight into the important problems, providing motivation for a new design through a portrayal of current practices. These scenarios form part of the CDR and are later linked to design scenarios, which describe how the newly developed interface is used. This

relationship—similarly defined between problem and design claims, allow developers to see connections between current practice and how their design affects and improves upon it. Designers also define design goals in terms of IRC critical parameters [31].

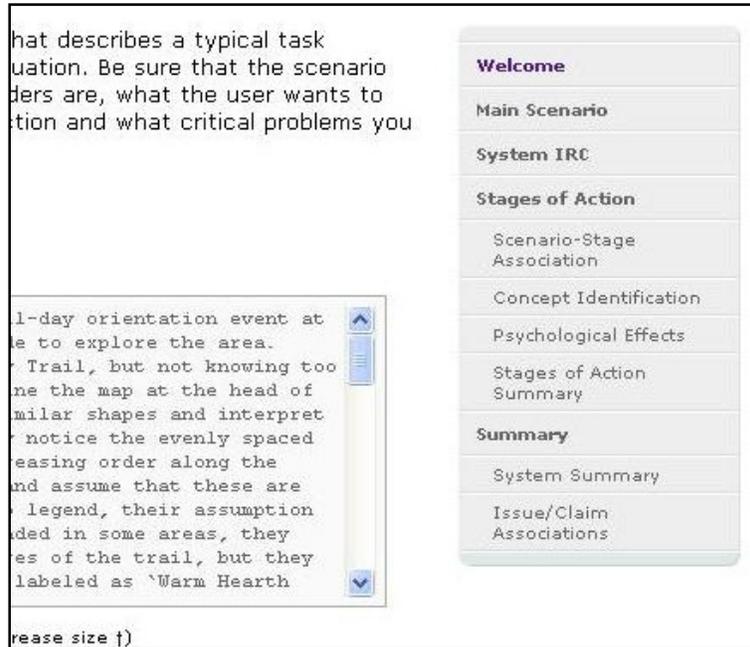


Figure 10. Part of the requirements analysis module. The progress/navigation bar is always visible on the right side of the screen to show designers where they are in the process.

4.1.6 Central design record module

Once developers have defined the problem situation and defined design goals in the requirements analysis module, the designer moves into the *central design record module*. The CDR module is the primary tool through which users develop and manage their designs. Users develop their CDRs by recording design scenarios and claims about the new system, relating them to the defined problem scenarios and claims, and iterating on them as they develop their interfaces. This module is more open-ended and less structured than the requirements analysis module to give users the freedom to change their CDRs as they develop and refine their designs.

Designers are first expected to create activity scenarios for each main task they have identified for their notification system. An activity scenario describes the high-level purpose and actions that are to be carried out in a main task [111]. Once the scenario is written, designers begin a claims analysis process for the scenario in which they gather claims for the activity scenario and establish relationships to the problem claims identified in the requirements analysis module. Just as in the requirements analysis module, the designer can either search for a claim or create a claim. The

process of gathering claims within the module supports further claims reuse for designs and encourages the creation of more claims when none are found. A similar process is again followed for information scenarios, scenarios depicting the information a user will encounter during the task, and interaction scenarios, scenarios describing the specific actions the user will take.

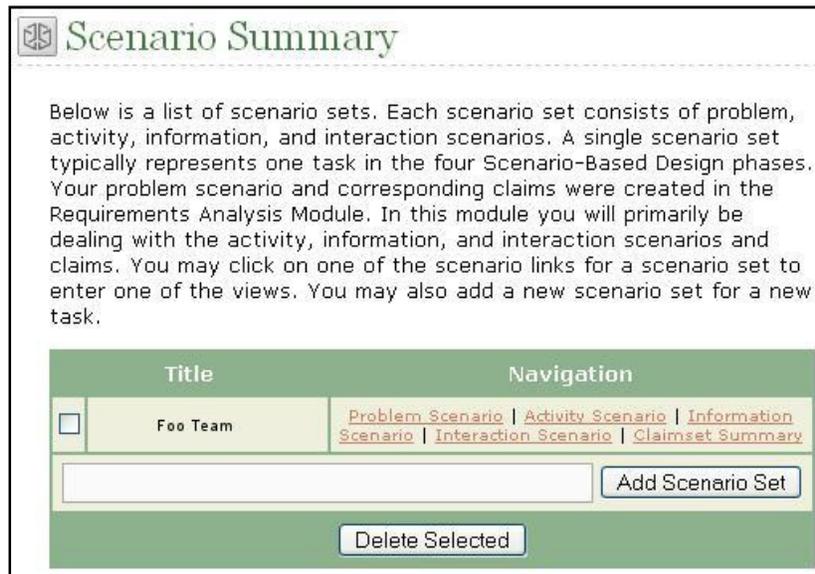


Figure 11. Part of the central design record module. Users create a set of scenarios and corresponding claims for each supported activity. Areas of the CDR module can be accessed in any order.

The ordering of steps in the CDR module is very fluid. A specific process, as opposed to the requirements analysis module, is not imposed upon the designer (See Figure 11). Many of the steps in the requirements analysis module, such as scenario decomposition, are excluded as it is expected that developers have internalized and understand these aspects of the design process at this point. Designers can switch between working on various parts of their design at any point in time. One can always choose to start creating a new task or to continue developing a previous task. This increases the flexibility of the module and permits the designer to revisit certain parts for redesign as a result of evaluation results.

The various portions of the module support the creation and maintenance of the CDR—a detailed representation of a design that can be used for evaluation purposes. For example, the breakdown of tasks in terms of the stages of action gives designers a chance to evaluate when they are nearing completion of a certain task, but also gives evaluators another perspective on how a certain task is being supported within a design. A breakdown of the design in terms of claims shows the links between prototype features and their corresponding tradeoffs encapsulated by the claims. Imposing

such structure on the design in the CDR module forms a gateway to facilitating communication. Stakeholders can discuss certain parts of the design with an established common ground that focuses on specific parts of the module, helping both designers and evaluators reach consensus.

4.1.7 Case study discussion

This section summarizes results from three case studies of design projects developed using LINK-UP over several development iterations are presented.

Table 1. Summary of strengths and weaknesses of LINK-UP derived from three development case studies.

| Key focus points | Case Observations by Design Phase | | | |
|--|--|---|---|---|
| | Requirements & Initial Design | Analytic evaluation | Redesign | Empirical evaluation |
| <i>IDE supports goal formation and facilitates design progress estimation</i> | (+) Generated IRC is close to expected values [case 2, case 3] | (+) Critical (IRC) parameters focus evaluation on non-trivial aspects of design [case 1, case 2] | (+) IRC parameters focus redesign efforts on critical design features [case 1, case 2] | (+) Evaluations focused on specific features verify hypotheses made in claims [all cases] |
| <i>IDE guides design representation development to support evaluation activities</i> | (+) CDR focuses on specific design features [all cases] (+) juxtaposing problem & design space allows careful reflection [case 1, case 2] | (+) Redesign efforts focused on features defined to be most important by developers [case 2, case 3] | (+) Claims analysis based on evaluation results guide redesign [case 1, case 3] (+) Untested or contested solutions can be deferred until empirical evaluation [case 1, case 2] | (+) link between design model and empirical results guides redesign [case 1, case 2] (-) general feedback not directly supported [all cases] |
| <i>IDE facilitates communication among stakeholders around design concerns</i> | (+) Paper prototype helped make design less abstract by connecting design rationale to interface. [case 1, case 3] | (+) CDR provides tangible design model to support discussion of interface between developers and evaluators [all cases] | → | (+) Conflicts uncovered in analytic evaluation resolved [case 1] |
| <i>IDE is flexible enough to support iterative design/evaluation activities</i> | (+) Scenario breakdown helps to find claims [case 3] (-) Requirements process constraining & tedious [case 1, case 2] (+) Claims search supported by critical parameters aids reuse [case 3] | (-) Hard to manage, comprehend large # of claims and scenarios in CDR [case 1, case 2] | (+) Claims format aided in analyzing design tradeoffs [case 1, case 3] | (-) functioning prototype (outside LINK-UP) provided most valued feedback [case 2, case 3] (+) validated claims can be updated with empirical data [all cases] |

Students in an undergraduate Human-Computer Interaction course at Virginia Tech used the IDE in a semester-long project to develop notification systems. Navigation-assisting notification systems with a focus on “off-the-desktop” systems were chosen as a general theme for all the projects. Each project group consisted of 4-5 students. They reported their progress in a series of project documents written

at each stage of development. These reports were designed to elicit feedback related to both the system they designed and the process and tools they used to design it. Six reports were written in total by each group, corresponding to requirements analysis, an initial design, an analytic evaluation, a redesign, an empirical evaluation, and a final concluding report. Both the strengths of this approach and areas for improvement are discussed, summarized in Table 1, with respect to the four focus points. Detailed results from the case study are available here [67].

The critical parameters of the notification systems, embodied in the system IRC values, proved to be a valuable guide in supporting the first focus point: IDEs need to support design goal formation and comparison between design goals and the resulting design artifacts. The case studies demonstrated the importance of integrating critical parameters into LINK-UP and guide reflection and iteration on the design through the CDR. They proved to both guide design activities and estimate design progress based on evaluation results.

The case studies also suggest the second focus point—IDE aids in design representation formation to focus evaluation activities and specific, incremental design improvements—is also supported. The CDR focused evaluation and redesign activities on specific aspects of the interface and encouraged careful consideration of current practices while developing the new system. Planning empirical evaluations partly around individual claims also supported the second focus point. Failures in the defined tests could be linked directly to design decisions—expressed in claims. Developers then knew where to focus prototype redesign efforts. The case studies demonstrate the value in supporting incremental improvements through the tight coupling of design representations with evaluation data.

LINK-UP, and its implementation of the CDR, also supports the third focus point—an IDE needs to support communication among stakeholders around the design representation’s development and evaluation. This work shows that by acting as a communication point between developers and evaluators, the CDR encourages discussion and evaluation of the interface design.

As the developers used the LINK-UP system itself, strengths and limitations of various parts of the IDE became apparent. Thus, the fourth focus point—IDE is flexible enough to support iterative design and evaluation activities—is partially supported. The relatively linear, guided process in LINK-UP, particularly in the requirements analysis module, was helpful to novice developers in identifying problem scenarios, claims and in defining design goals. However, all groups found this process to be tedious and restrictive to varying degrees. Furthermore, several groups noted the

difficulty they had in managing and reviewing the large number of scenarios and claims through LINK-UP. The size of the CDR may cause other issues in the design process such as in analytic evaluations where expert evaluators may have problems reviewing and making sense of all of the CDR information.

4.1.8 Summary of study findings

Numerous usability methodologies and techniques exist to support the development of computer systems and interfaces. However, it is not always clear to novice developers how to use these different processes and techniques in a coherent design process. Through the case studies this research demonstrates that a design knowledge IDE centered on the central design record can help novice developers make connections between requirements data, design representations and evaluation data and better understand how to leverage that information to incrementally improve designs in an iterative usability engineering process. It is also shown that the use of CDR supports the application and verification of reusable design knowledge for novice developers.

Based on these results, the following guidelines are derived for design knowledge integrated development environments for use in educational settings that incorporate a module patterned on the CDR:

- Persistent design representations should support multiple or ‘current’ perspectives to direct development efforts on salient design concerns and streamline development processes.
- Design rationale should be tightly coupled to evaluation data to show novice developers where to direct redesign efforts and support validation of design knowledge for future reuse.
- Design representations need to be easily understandable, with goal states stated in unambiguous terms, perhaps through critical parameters, to support collaborations among novice developers and other project stakeholders.
- Design knowledge IDEs should support, but not require, guided processes to aid in knowledge capture, knowledge use and goal formation so that they support developers with varying levels of knowledge and expertise.

4.2 Developing the XSBD process

The growing importance of computing systems in everyone’s daily lives has made software development an inherently multidisciplinary endeavor [98, 101]. My previous work with LINK-UP

and the CDR demonstrated the potential for a reuse-enabled integrated design environment in guiding novice developers through the interface design and evaluation process [62, 63, 67]. However, that work was not sufficient to enable usability engineers to work effectively within a multidisciplinary development team. In practice, one needs to determine how usability specialists can work within a team to develop systems in ways that can best leverage the perspectives, practices and knowledge bases of people from different areas. Software engineers, who focus more on the design and implementation of software systems, and usability engineers who focus more on the interface design for end-users, are two areas of design that have not traditionally worked well together. This is especially true of agile development methodologies, which have emerged in the last decade as a way to address problems associated with software complexity and constantly changing requirements. This work leverages the lessons learned in my previous work [62, 63, 67], including the use of design representations such as the CDR, and how tools such as LINK-UP should be designed, to develop practices and tools to support agile, multidisciplinary teams in developing usable systems in an efficient manner [64].

The differing goals and motivations of practitioners in software and usability engineering combined with the myriad techniques and methodologies in each leads to tensions that need to be addressed in any development process that draws on both. In this work, we probe these tensions by looking at prominent development practices in agile software development and usability engineering, namely extreme programming (XP) and scenario-based design (SBD) respectively, and explore how they can work together in developing usable software systems efficiently. Scenario-based design and extreme programming are built on similar foundations. Both support iterative development, are human-centered and emphasize team coordination and communication. However tensions between the two approaches need to be addressed for them to work together effectively. The XSBD process supports the best practices of both processes while mitigating the tensions between them. The key features of the process are defined below. Development oriented practices such as pair programming, unit testing, code refactoring and continuous integration are unchanged. This work will focus on how to integrate SBD and XP so that usability goals can be met without violating key agile development tenets as laid out in the Agile Manifesto [13].

This section is adapted from my CHI 2006 doctoral consortium paper: *Embracing Agile Development of Usability Software Systems* and my Agile 2007 conference paper: *Towards Extreme(ly) Usable Software: Exploring Tensions Between Usability and Agile Software Development* [61, 64].

4.2.1 Interface architecture design

Development occurs in short iterations (2-4 weeks), beginning with an abbreviated requirements analysis process (See Figure 12). In this process, the interface design—embodied in the CDR—is developed using a depth-first approach and is continuously improved and evaluated throughout the development process. This feedback loop is achieved by defining critical parameters/usability goals at the start of the process, and then using the CDR to identify critical areas of the design that need to be evaluated. Claims describe these critical areas—highlighting high-risk factors and assumptions that are tested at the end of each iteration.

Software development and usability engineering will generally occur in parallel. A successful development effort hinges on proper balance between development concerns of different stakeholder groups. The CDR acts as a shared design representation that allows software developers and usability engineers to have a shared understanding of the design that allows them to make appropriate design tradeoffs based on constraints (e.g. usability problems, technical limitations, scheduling problems). It will be a focus point of review/planning meetings that occur at the beginning of each iteration and release cycle. The CDR is a form of design rationale which captures and makes explicit the information related to why certain design decisions were made. By making the design decisions explicit, design rationale approaches can help designers make more reasoned, systematic decisions that can be captured and potentially reused [20]. Such approaches have been introduced not only in HCI but also in software engineering. For example, Salasin introduced the concept of the ‘design record’ as a way to capture and represent system information and rationale for design decisions [112].

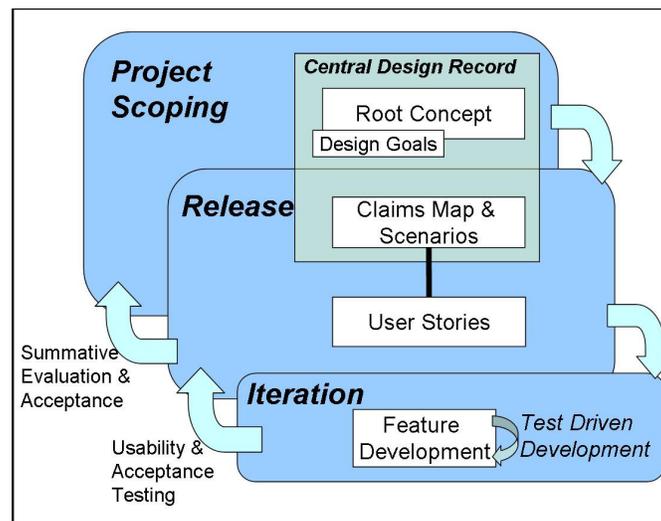


Figure 12. Key steps in XSB D process. Design artifacts are shown in white.

The *central design record* (CDR) is the main interface design representation in this process and is conceptually similar to previous efforts [62, 63, 67]. It consists of the set of scenarios describing different usage situations, interrelated claims describing specific features in the interfaces, and design goals. Design goals are stated in terms of critical parameters which are measures of performance used to determine how well a design serves its purpose [91, 92]. It is used to guide all stages of interface design in the XSBD process. Task analysis and story elicitation typical occurs first and leads to scenario development but the reverse can also happen. The CDR allows developers to systematically improve the interface during the development process because it stores the rationale for the design decisions within an organized set of claims called the claims map [130, 132] (Figure 12). This makes design decisions explicit and highlights important relationships between different parts of the interface.

The challenge is to develop, maintain and make use of the CDR within the tight time constraints of the XP development cycle. In the XSBD process, the CDR will be expanded as development proceeds incrementally. Developers will maintain a consistent overall view of the interaction architecture and the activities it supports by continuously reviewing and updating it throughout each iteration.

4.2.2 Collaboration through the CDR

As a record of design decisions made to the interface, the CDR also acts as a communication point between and among developers, evaluators and clients. Previous studies have shown that the CDR helps developers communicate design decisions to other stakeholders and facilitates the resolution of design issues uncovered in usability evaluations [62, 63, 67]. Scenarios provide easy-to-understand narrative descriptions of how users will interact with the system. Critical design decisions and tradeoffs are encapsulated in the claims, and allow developers to quickly compare different design options. It also allows them to justify design decisions to other stakeholders and to better plan for and direct meetings with clients or users.

4.2.3 Usability evaluations through the CDR

The other addition we made to the existing XP framework is usability evaluations. XP includes unit testing which verifies the functional accuracy of the code, and acceptance testing which proves to customers that the system works as agreed upon. Usability testing will verify that the system is easy and intuitive to end users and is critical to closing the feedback loop that will support continual improvement of the user interface. Although there may be some overlap between acceptance and usability testing, the focus of each is distinct and equally important. Scenario-based design, the CDR,

and related techniques from usability engineering provide the framework and guidance necessary to support usability evaluations.

Claims in the CDR are used to analyze and reason about specific interface features. Claim downsides or upsides can be validated through usability evaluations and help developers identify usability problems. By tracking which claims correspond to the stories currently being developed and looking at their interrelationships in the claims maps, developers can plan targeted evaluations at the end of each iteration. They also leverage light-weight usability evaluation methods such as expert walkthroughs and heuristic evaluations to quickly evaluate the interface. This process complements the XP practice of test-driven development to maintain functional correctness of the code. In this case, the usability of the design can be validated at regular intervals to prevent entropy in the overall interaction design as the system is incrementally developed.

4.2.4 Design case studies

Two design case studies were conducted to demonstrate how the XSBD process addresses the key questions detailed in Chapter 1. The developers were four undergraduate students receiving research or independent study credit. Three people, including the authors of this paper, acted as managers and oversaw each development project. The students were introduced to the XSBD process over the course of several weeks at the beginning of the semester. The remaining 10 weeks were devoted to development.

Each group used the XSBD process to develop their respective systems. Development proceeded over the course of five two week iterations, representing a single release cycle. Project information and CDR documentation was stored on a wiki that the developers and clients could access at any points in the project [68]. The development environment was made as realistic as possible but there were several limitations. First, the developers and clients were all located at the Blacksburg Campus at Virginia Tech but developers and clients were not collocated in a single office environment due to work and academic obligations. However, developers at least held weekly meetings with their clients and remained in constant email contact. In addition, pair programming was not used consistently throughout the semester due to conflicting schedules. They were advised to conduct code reviews together when they had to work separately. Project managers did a code walkthrough with one or the other member of a team (including unit tests) at the end of each iteration to verify that both developers in each team understood the code. Detailed results from the case studies can be found in my Agile 2007 conference paper [64].

Evolution of the CDR

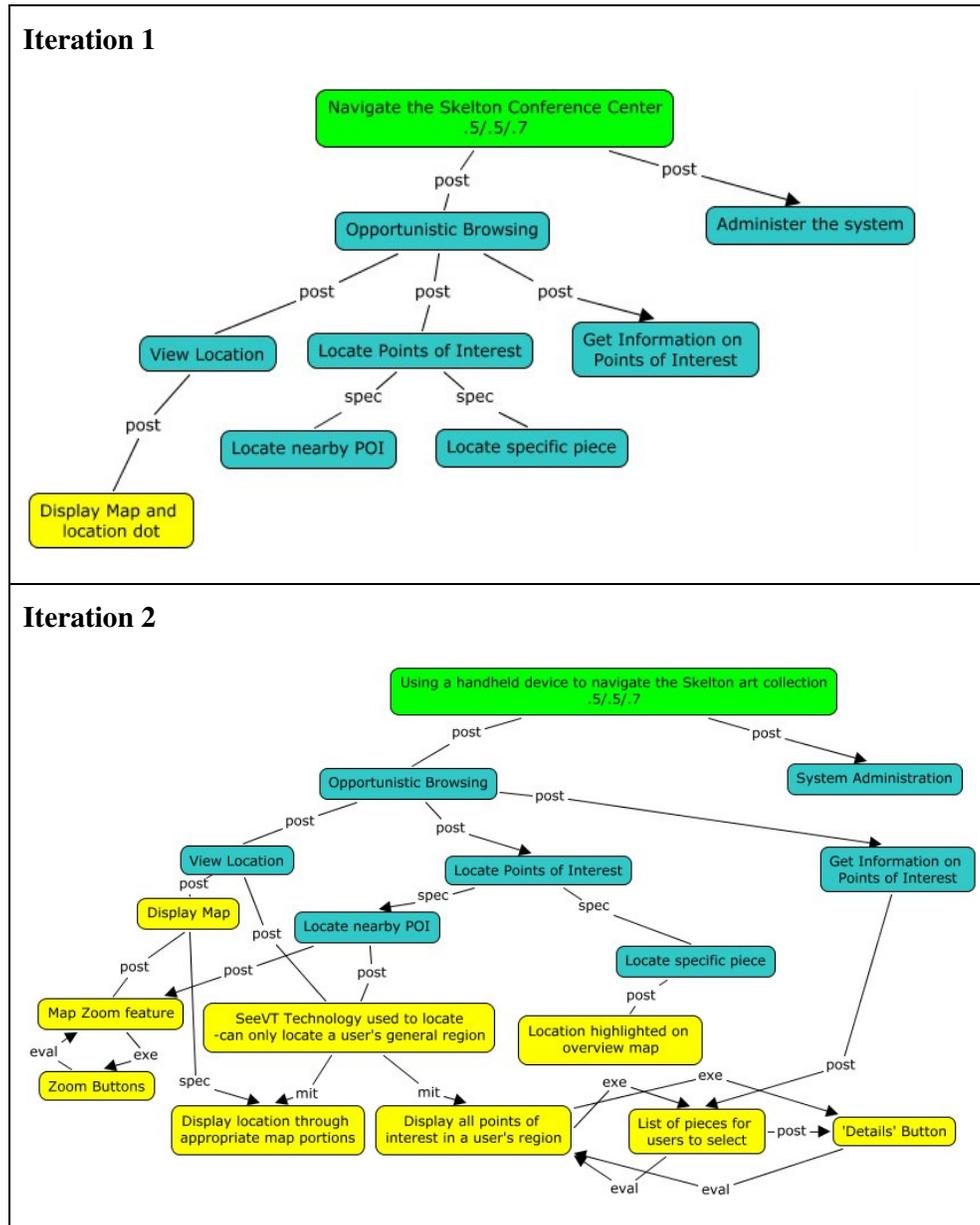


Figure 13. Example claims map. The root concept claim (green) is linked to activity claims (blue) which are linked to implementation claims (yellow). Note the progressive growth through the iterations. (Note that only claim features are included in the diagrams)

The CDR was used to maintain a coherent, consistent and understandable interaction architecture within the incremental agile development process. It provided a broad overview of the design and specific details when needed to guide usability evaluations. The relationship between the CDR and the stories being developed enabled the developers to make key decisions in the development process.

Each project began with a release planning meeting involving the developers and their respective clients. System conceptualization, stakeholder identification, and high-level design goal formation were done in this initial planning meeting and iteration.

After developing an initial list of design scenarios, specific claims are developed that correspond to the tasks that each system needs to support. The claims are arranged into a *claims map*, which shows how the different tasks and interface features are related. The claims map starts with a *root concept claim*, which describes the system being developed in addition to important tradeoffs that need to be considered. These tradeoffs can relate to a mix of technological, contextual and usability issues.

Activity claims radiate out from the root concept claim and correspond to the specific tasks the system should enable. More specific claims are then linked to the activity claims which describe exactly how those tasks are supported. These specific claims can describe how information is displayed to the user and what interactions are supported. They generally correspond directly to one of the stories. In the first iteration, the claims map primarily consists of the root concept claim and the activity claims. At each iteration, developers implement a small subset of the total functionality of the system. Similarly, the CDR grows incrementally as this functionality is developed (See Figure 13). The organization of the CDR shows the design of the system at multiple specificities—from the conceptual level to the task level to the interface level. Maintaining this organization gives developers a constant overview of the most important features of the interface, the tasks they support, and how they are interrelated. This claims map allows developers to refactor the interface to reflect new usage scenarios and changes in response to client feedback and evaluations. A detailed discussion of the relationships used in the claims maps is available here [130]. The claims map becomes increasingly complex as design proceeds, but the root concept claim and activity claims are relatively stable throughout the development process although they can change from iteration to iteration.

Connecting SBD and XP

In the XP+SBD process, usability engineering and system development occurs in a single unified development process. In larger teams, different developers may focus more on usability or development depending on their expertise, similar to the development process followed by Lynn Miller and the team at Alias [84, 127]. The important point is that both software development and usability issues are considered concurrently throughout the development process. This ensures that the entire team understands both aspects of development and how they interact. One group demonstrated

how different concerns can overlap when they were forced to redesign their interface to deal with a limitation of the location-detecting technology they were using.

Evaluating the Interface

Evaluating the usability of the interface, unlike unit testing, is difficult to completely automate. Evaluations can result in a large amount of quantitative and qualitative data that touch on many different aspects of usability including ease of use, learnability and overall satisfaction. They also require an actual person or persons to conduct the testing. As a result, the XP+SBD process advocates the use of light-weight analytic evaluations, which occur at the end of each iteration followed by more in-depth usability evaluations at the end of each release cycle. The CDR is used to guide these evaluations by helping developers determine what areas of the interface—described in claims—to evaluate at the end of each iteration and what effect redesigning some part of the interface will have on other parts of the system. The only time where a large amount of time was spent running an evaluation was at the end of the semester when a comprehensive usability evaluation of the entire system was conducted. This is needed at the end of release cycles to validate overall usability and uncover additional usability problems.

Communicating Design Rationale

The different parts of the CDR facilitated communication of design rationale among project stakeholders. The student developers were not explicitly told what parts of the CDR to share with their clients but they were encouraged to share materials they thought could facilitate communication. High level design goals and scenarios proved to be useful for conveying information to clients. Claims were not explicitly shared with clients. Developers relied more on verbal communication of design concerns and issues.

4.2.5 Discussion of results

Table 2 summarizes how the XSBD approach addresses the tensions between agile methods and usability. The incremental development illustrated by the CDR showed how it was possible to develop a coherent and consistent interface design representation by continuously reviewing the claims map and refactoring the interface when the need arose. This allowed agile developers to maintain a consistent, incremental release cycle throughout the project. However, it can be difficult to determine how complete a design representation should be. The developers in both projects largely relied on their own judgment as to what parts of the interface should be represented and which did not need to be. There is a tradeoff between completeness and manageability of the representation that developers

will have to balance when using a design representation like the CDR. Critical parameters were shown to be a useful guide for defining overall project goals and measuring progress. However critical parameters can be difficult to define or measure depending on the type of system being developed. The developers relied mostly on rough estimates for the IRC values which still proved useful in communicating design goals among themselves and their clients.

The developers' use of claims and claims map showed how useful it can be to maintain a list of interface design decisions and the potential tradeoffs they entail. These helped them to see exactly what parts of the interface needed to be tested in the current iteration and which could be deferred or ignored. Many of the evaluations were in fact folded into regular client meetings.

Table 2. Table shows how the XSBD approach addresses tensions between agile methods and usability.

| |
|---|
| <p>How can developers design consistent and coherent interface architectures within an incremental agile development framework?</p> |
| <p><i>Incremental development of an interface supported by a design representation like the CDR</i> + Can help developers maintain consistent and cohesive interaction design through the continual evaluation of design rationale and targeted interface improvements. + Does not limit or excessively delay incremental software delivery - Can be difficult to determine when a design representation is sufficiently complete <i>Using critical parameters like IRC values to guide interface development</i> + Can be used to measure design success through repeated evaluation of explicit metrics - Critical parameters may be difficult to define and measure</p> |
| <p>How can usability evaluations be streamlined so they better fit in accelerated development cycles while still providing useful results?</p> |
| <p><i>Maintaining an organized list of design tradeoffs, to guide lightweight usability evaluations</i> + Can allow designers to target specific areas of the interface to evaluate, thereby saving effort and reducing the need to reevaluate parts of the interface in later iterations - Requires additional effort by developers to plan and run usability evaluations</p> |
| <p>How can project members support communication and cooperation between designers, customers, users and other stakeholders who have different backgrounds and expertise?</p> |
| <p><i>Shared design representation showing high-level and detailed views of the interaction design</i> + Allow different stakeholder groups with different backgrounds to understand and give feedback about the design. + Makes interplay between usability and agile development explicit and understandable + Can focus planning by reminding stakeholders of key design decisions and concerns - Requires designers to actively maintain links between agile and usability artifacts</p> |

The cases also show that a design representation consisting of easy to understand artifacts that make explicit the connections between the different concerns of the stakeholder groups can support communication and cooperation. Communication and buy-in is vital for these different groups to work together effectively. This representation allowed different stakeholders to have a shared understanding of the overall design and to make informed tradeoffs when conflicts came up. However, maintaining this kind of representation does take some effort.

4.2.6 Summary of study findings

To that end, this work has focused on finding ways for agile software developers and usability engineers to work together more effectively by addressing the conflicts between extreme programming and scenario-based design. We end with four guidelines for usability specialists and agile practitioners that can be derived from this work.

- Share design documents and artifacts when possible. Maintaining communication among team members is vital. Sharing these artifacts can augment face-to-face communications and allow developers to make informed decisions about design tradeoffs.
- Strive for continuous interface improvement. Incremental additions to an interface can gradually erode overall usability. Always be aware of possible improvements and do not be afraid to test new ideas. A larger number of smaller, more focused usability studies can result in a similar (or better) level of understanding as a small number of large tests—and it better fits with the agile philosophy.
- Integrate usability into day to day development tasks. Continuous improvements require continuous user feedback. Conduct informal evaluations when more complete usability evaluations are not feasible. Have clients or other team members look over new interface features. Such data can be valuable when you know who you're designing for, what data you're collecting and why you're collecting it.
- Avoid having team members overspecialize in one area. Team cohesiveness is important to maintain velocity. Members with separate focus areas/expertise should have an understanding of each other's specialties to prevent misunderstandings and wasted work.

4.3 Chapter summary

This chapter summarized my initial work in developing the XSBD process. It focused on how I first developed the CDR to help novice developers learn and apply usability concepts while developing a system. I then adapted the CDR as a way to not only capture the interface design, but to also aid in collaboration and communication between usability professionals and agile software developers. I show, through several student led development efforts, how the CDR can help developers maintain a coherent interface and interaction architecture within an agile development environment. However, there were still a number of issues that needed to be resolved for XSBD to be used in practice effectively. First, it was shown that different aspects of the CDR such as claims maps are not easily shared with different stakeholders. This indicated that the CDR as it was currently constructed may need to be changed to better support communication between different stakeholders. Second, results showed that it was sometimes difficult to manage the CDR while trying to maintain a tight development schedule. There was an outstanding need to streamline and adapt aspects of the CDR to reflect the needs and concerns of different stakeholders so development decisions and general project management could proceed more smoothly. Finally, one limitation of the studies was that they were student-led development efforts that were not subject to many of the challenges and circumstances faced by industry professionals.

It was anticipated that evaluating the approach in practice with experienced developers would provide richer, more realistic data that would substantiate the utility of XSBD to agile practitioners. To that end, I partnered with Meridium—a software and services company—to evaluate and improve the XSBD process using an action research methodology. These case studies are described in Chapter 6. However, I first provide an overview of the XSBD process as it currently exists in Chapter 5.

5 eXtreme Scenario-Based Design

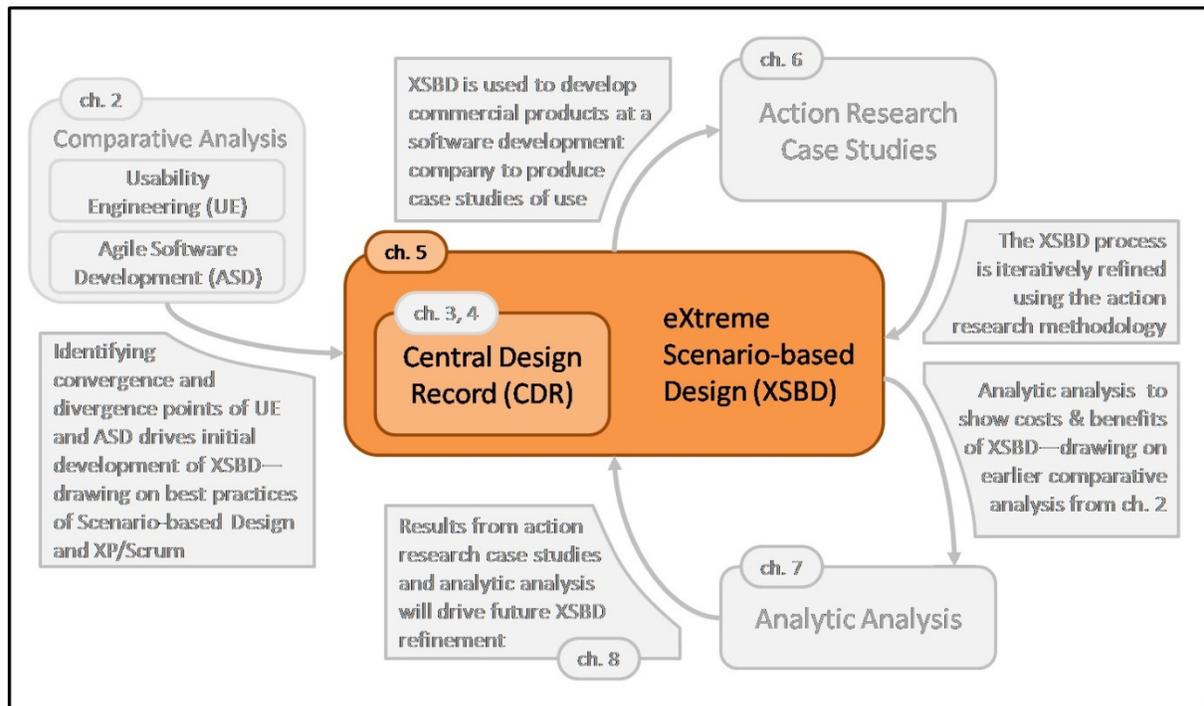


Figure 14. Overview of solution approach. Chapter 5 will provide a summary of the XSBD process as it currently exists. Details of how it was used in practice will be described in Chapter 6.

This chapter will provide a complete overview of the XSBD process that resulted from real-world deployment and incremental revisions (See Figure 14). It is partially adapted from my Agile 2009 conference paper: *Examining the Foundations of Agile Usability with eXtreme Scenario-based Design* [65]. The process description in this chapter is provided to partially address the last three issues listed in section 1.1 *Challenges facing an integrated approach*. It will describe how agile development and usability design concerns are balanced through proper consideration of design tradeoffs. It will also detail how agile development and usability engineering remains synchronized throughout the design effort through the use of the CDR, and shared communication and process planning practices. It will describe the optimized CDR I developed for use in practice. The CDR as it is used in XSBD teams is critical to supporting the coherent development of the user interface within the incremental agile development framework. It also supports communication of design rationale to other team members and helps them make balanced design decisions. Finally, this chapter will describe how the XSBD process was developed so that usability engineering practices were embedded within agile planning and development processes and why this was done. A detailed discussion of how this process was substantiated and refined in practice using an action research case study methodology will be described in Chapter 6.

This chapter will first provide a brief overview of the different components of the process and how they are related to each other. Second, it will describe the different roles in the process—describing what each role is responsible for and why. Third, it will detail the modified CDR that was developed over the course of the two action research case studies at Meridium. Fourth, it will describe how the CDR is used in the XSBD process to support incremental UI development while maintaining adherence to high-level design goals. Fifth, it will describe how development and usability remains synchronized so that interface design and development can proceed in parallel. Finally, it will describe how the CDR is used to make balanced design decisions throughout the development process—taking into account usability, development and product management concerns.

5.1 Overview of XSBD process

This section will give a brief overview of the XSBD process and the characteristics of projects that use it.

5.1.1 XSBD in brief

The primary purpose of the XSBD process is to allow an agile team—which includes a dedicated usability engineer—to efficiently develop a software system that meets the project’s usability and development goals. XSBD draws on a number of practices from SBD which are adjusted to work in an incremental agile framework. This agile framework is based on key practices and processes from XP and Scrum—two leading agile methodologies.

There are five primary roles in the XSBD process: the usability engineer, developer, end user representative, customer representative and project manager. The usability engineer and end user representative are primarily concerned with the user interface design. The developer and customer representative are concerned with the development of the system. The project manager is charged with maintaining and tracking the project schedule. In the end, the efforts of the XSBD team will be to meet the needs of the customer representative as the purpose of developing the system is to optimize value delivered to the client.

The CDR is the primary design representation used in the XSBD process to manage the user interface design. It is maintained by the usability engineer, but parts of it are shared with the other members of the XSBD team to communicate information such as design rationale and user interface changes resulting from usability evaluations. It consists of three major parts: high level design goals and

visioning, design artifacts in the form of scenarios, claims and design prototypes, and usability evaluations. Directly linking high-level goals to design decisions and validating them through usability evaluations is an efficient way to ensure that the system meets usability goals.

In the XSBD process there are separate but synchronized usability and development tracks similar to the parallel approach described by Millar and Sy [84, 127]. The usability engineer works one iteration ahead of the developers so designs can be delivered for the developers to start implementing at the start of each iteration. Through careful coordination of the development and usability tracks and use of the CDR, the XSBD team can optimize its velocity while still developing a system that meets high-level design goals—enabling balanced design decisions that incorporate the oftentimes conflicting needs of the developers and usability engineer.

5.1.2 Characteristics of an XSBD project

The XSBD process has been developed for use in agile projects that have a non-trivial usability component. This means that the proposed user interface will comprise a significant amount of the development effort and that usability is one of the key high-level goals of the system being developed. Optimally, the entire team will be collocated although it is possible for a distributed team to use the XSBD approach (See Section 6.3). In practice, each member should be spending at least 50% of their time on the project. In addition, the XSBD process has been used in smaller development projects with approximately 10 people including 2-4 developers and 1 usability engineer.

The remainder of the chapter will give a basic overview of the XSBD process based on these characteristics. Variations on this basic XSBD process will be described in more detail in chapter 6.

5.2 Roles in the XSBD process

This section will describe the primary roles in the XSBD process, explaining what their responsibilities are and what other roles they interact with. It is possible for a single person to take on more than one role in a project depending on the size and nature of the project. This will be described in more detail in Chapter 6 where the XSBD case studies are discussed.

Usability engineer. The primary goal of the usability engineer is to ensure that the system meets end user needs and meets high-level usability goals defined for the project in terms of effectiveness, efficiency, satisfaction, ease of learning, memorability and minimization of errors [95]. The usability engineer takes functional requirements and translates them into a user interface that meets usability

goals through a combination of requirements analysis, interface design, prototyping and evaluations. These goals, design decisions, and evaluations are planned and managed through the CDR. The usability engineer is primarily responsible for managing the CDR and communicating the design aspects it captures to other team members as necessary. The usability engineer, supported by the CDR, acts as a bridge between the other user roles—especially the end user representative, customer representative and the software developer. He or she works with and communicates with the customer representative to ensure the system meets the business and functional requirements of the system and works with the end user representative to ensure that it meets end user needs. The usability engineer also works with the developers to ensure that the implementation matches the user interface design and to manage changes that need to be made due to changing requirements and technical limitations.

End user representative. The end user representative is tasked with working with the usability engineer to ensure that the product being developed meets end user needs. Unlike the customer representative—who is responsible for validating that the software being developed is functionally correct—the end user representative is concerned with ensuring that the functionality being developed is usable for end users. The end user representative will optimally be working at the client company, and will be one of the users of the system being developed. Unlike the customer representative, he or she has firsthand knowledge of the work processes the software system will support and is intimately familiar with the context in which it will be used. The end user representative works most closely with the usability engineer and provides important information during requirements gathering activities. He or she provides feedback to the usability engineer on prototypes and the implemented system. The end user representative is also tasked with helping to recruit other end users at the client company to participate in user evaluations led by the usability engineer.

Software developer. The primary goal of the software developer is to implement the software system by translating user interface designs and functional requirements to working code. The software developer does this in an incremental development cycle, taking designs and requirements from the usability engineer and customer representative, developing the code to instantiate them, and fixing bugs and refactoring the code as necessary. The developer is primarily responsible for the engineering methods and implementation design artifacts. This will include things such as the software architecture design and agile practices such as test driven development and code refactoring [11, 12, 115, 116]. The software developer works closely with the customer representative to ensure that features are being developed as requested. He or she also works closely with the usability engineer to ensure that the user interface designs are being implemented properly, to address conflicts that arise

between the designs and the implementation, and to make changes based on evaluations run by the usability engineer.

Customer representative. The customer representative is tasked with ensuring optimization of return on investment and alignment with the client company's goals. Unlike the end user representative, the customer representative is primarily concerned with ensuring the functional correctness of the system. The customer representative does this by defining the requirements for the project and reviewing development progress to ensure functional correctness. This includes specific tasks such as determining what features the system must have and what their priorities are, participating in regular functional review meetings and being available to answer questions about features throughout the development process. Like in XP, the customer representative is optimally someone from the client company that the XSBD team is developing the software for and is embedded with the XSBD team on site [12]. The customer representative can also be a subject matter expert working at the software development company who acts as a liaison to the client company. The customer representative works most closely with the software developer and project manager—communicating functional needs and determining whether they are met in the finished product.

Other roles. There are several other roles that were identified in the course of implementing the XSBD process through the action research case studies at Meridium (Chapter 6). The project manager is tasked with the overall management of the software project and ensuring the project is completed in a timely manner and meets stakeholder needs in terms of functionality, usability and performance. The project manager maintains the project schedule, ensures that each iteration proceeds as planned and makes sure that all team member tasks are in the schedule and are completed at the end of each iteration. Quality assurance specialists are tasked with verifying the functional correctness of the software system. In some agile teams that use practices like XP, this is done by the software developer using practices like test driven development [12]. However, quality assurance specialists are needed in larger software organizations and are specially trained to test software systems and ensure that functional integrity is maintained across different releases and upgrades. The second role that was found was documenters. The documenter's role is to write the associated documentation for the software being developed. This includes installation instructions and instructions in the use of the software.

5.3 The Central Design Record

The Central Design Record (CDR) is the primary design representation used in the XSBD to help ensure that the user interface design meets the needs of the end users and customer. A description of how I developed the CDR as part of my early research efforts in usability engineering and design rationale is provided in Chapter 3 and Chapter 4. It is primarily maintained by the usability engineer but parts of it are shared with the other members of the XSBD team to communicate information such as design rationale and user interface changes resulting from usability evaluations. I have streamlined this version of the CDR so it can more effectively be used in agile processes by reducing the cost of maintaining it while keeping its most important characteristics [61, 62, 63, 64]. A detailed description of how and why it was streamlined will be described in Chapter 6.

5.3.1 Overview of the CDR

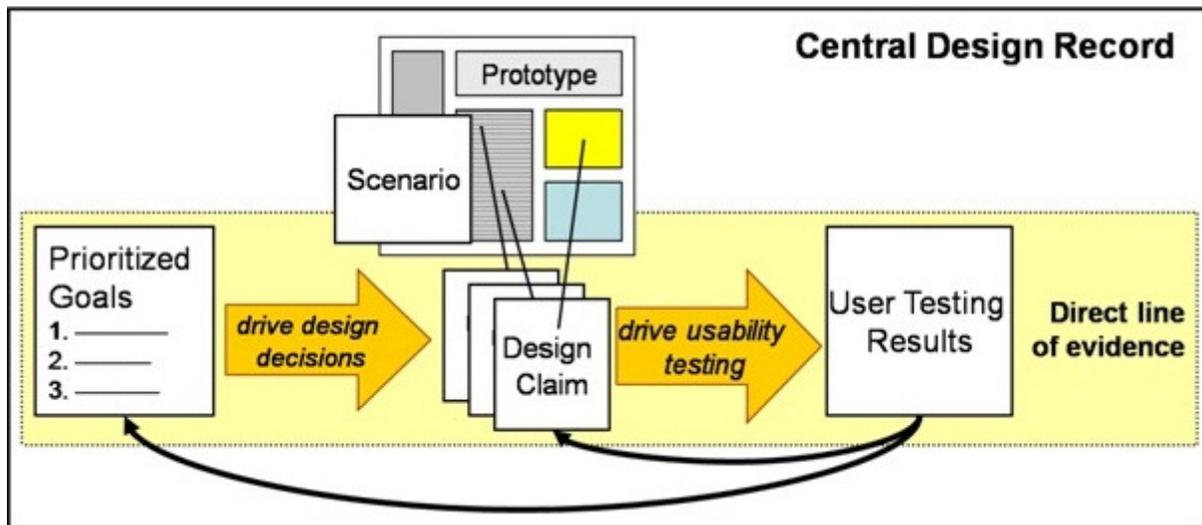


Figure 15. Streamlined version of the Central Design Record developed for use in the XSBD process.

Within the XSBD process, the CDR is the primary design artifact used to guide the development of the user interface. It allows the usability engineer to work within the incremental agile development cycle while maintaining the high-level vision of the interface. It helps the usability engineer to execute usability evaluations that fit within the agile framework while validating that the user interface is usable and meets the high-level goals of the system. Finally, it supports communication of design rationale to other XSBD team members and helps them make more balanced design decisions.

The CDR consists primarily of three parts (See Figure 15):

- 1) Prioritized project goals along with user descriptions and a vision statement which provides high-level guidance as to what will be developed.
- 2) Scenarios, claims and design prototypes that capture the design and critical design decisions.
- 3) User testing results that validate design decisions and ensure that the design meets the project goals.

Each part and their role in the CDR is described below.

5.3.2 Project vision and prioritized project goals

It is important for the entire team to have an understanding of what system is being developed, for whom the system is being developed and what the high-level goals of the system are. This will help the team stay focused on the system being developed, and help them to collectively make appropriate design decisions and tradeoffs as development proceeds.

Vision document. A vision document prepared with the customer representative will summarize what system is being developed and why. It will help the development team gain a common understanding of the system being developed [111]. This project vision is meant to be internalized by all XSBD team members and should be relatively static. This document may be longer and more detailed depending on the needs of the customer representative and contract negotiations between the development organization and the client company.

User descriptions. The XSBD team and the usability engineer in particular needs to understand who the end users of the system will be, what their needs are and what context the system will be used in [111]. The usability engineer will develop user descriptions for all the important end users of the system based on requirements gathering activities and information gathered from the end user representative. The usability engineer will design the system to meet these user needs and will rely on them to communicate design decisions to other team members. Like in SBD, In XSBD, the user descriptions will include things such as user goals, user characteristics and environmental/contextual factors. Alternately, personas—rich narrative descriptive of hypothetical users—could be developed to help other team members understand and emphasize with end users [38].

Prioritized goals. Prioritized goals are a critical piece of the CDR and serve as part of the link that comprises the goals of the system, the design that instantiates those goals, and the evaluations that validate that the design meets those goals. They consist of both usability design goals, and development-oriented goals including performance targets and delivery target dates. They are

prioritized based on the needs of the customer. These will help the team to make design tradeoffs throughout the development of the system. These goals are typically stated as a single sentence in plain English. For example, the prioritized goals for a self-checkout system at a grocery store might be:

Design goals for self-checkout system

- User shall be able to quickly use the system to checkout with minimal frustration.
- User shall be able to quickly learn about and use the system.
- System shall prevent theft of goods.

In this simple example, the customer decided that since most people are honest, it was more important for the system to be usable than to be secured against theft of goods. Specific metrics are associated with each of the goals were possible to make it easier to validate that a particular goal has been met. For usability-related goals, these could be in the form of critical parameters—which are figures of merit that measure the performance of the system and aid in comparison to other similar systems [91, 92]. Usability-related metrics are developed and maintained by the usability engineer while development-related metrics are developed and maintained by the developers. Initially, only the prioritized goals themselves are shared and understood throughout the team to give all team members an idea of what the goals are and their importance in relation to each other. Specific metrics are shared only as necessary. As an example, specific metrics for the first goal might be:

- User shall be able to quickly use the system to checkout with minimal frustration.
- Checkout of 10 items shall take no more than 2 minutes.
- Checkout of 10 items shall average .5 errors.

5.3.3 Design artifacts

The primary user interface design artifacts that are part of the CDR are the *scenarios*—which describe common workflows, *prototypes*—which are typically mockup representations of the design, and *claims*—which highlight design tradeoffs of critical design features. These artifacts are generally developed within a single iteration and so are directly related to the features or stories (using XP terminology) that the developers manage while scheduling their development tasks. A more detailed description of how this is done is detailed in Section 5.4.

Scenarios. Scenarios as used in XSBD are narratives describing the activities of one or more persons and include information about user goals, actions and reactions [111]. Traditional scenario-based design includes four types of scenarios. Problem scenarios describe current practices that have implications for the design of the system. Activity scenarios focus on the actions and goals of the user as he or she uses the new system. Interaction scenarios describe detailed information about how the user interacts with the user interface (e.g. specific mouse/keyboard actions). Information scenarios describe how information is presented to the user (e.g. screen layouts, feedback). The CDR as used in the XSBD process relies exclusively on problem and activity scenarios as these can give other team members a good understanding of how the system will be used and how it addresses user needs. This is especially useful as it provides a cohesive narrative description of how the features in an iteration are interrelated. Specific information related to interactions and information presented to the user will be communicated through screen mockups and face-to-face communications. An example activity scenario for the automated checkout system example is presented in Figure 16:

Mike is preparing for a dinner party and has rushed to the SuperMax Grocery to purchase several items. He sees that most of the checkout lines are long but notices that the new automated checkout line is free. Although he has never used the system before he decides it will be quicker to use than waiting in one of the other lines. The instructions on the screen tell him to scan his first item and place it in the bag. He does so after a few seconds of looking to see where the barcode is. After placing the item in the bag, the system tells him to scan the next item and place it in the bag. He repeats this for each of the items with increasing efficiency. After scanning the last item, he indicates that he is finished and would like to pay. He chooses to pay by credit card and then uses the automated credit reader with ease, as he has used similar scanners before. After paying the system informs him that the transaction is complete and reminds him to take his bags. He grabs his bags and leaves the store.

Figure 16. Example scenario.

Design prototypes. Design prototypes, typically in the form of non-functioning mockups, are important in that they are one of the primary mechanisms through which the usability engineer communicates the user interface design to the developers. Coupled with the scenarios and regular face-to-face communication, they allow the developer to translate the user interface design to a functioning system.

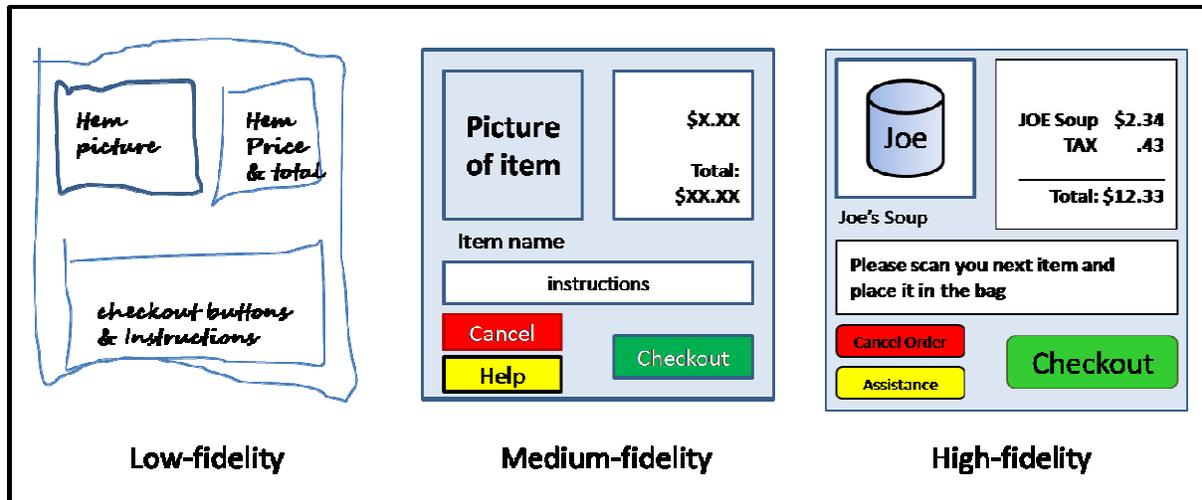


Figure 17. Medium-fidelity prototypes provide more details than a low-fidelity prototype but does not provide exact positioning/color information.

Design prototypes are usually ‘medium-fidelity’—meaning that they are detailed enough to allow the developer to implement while requiring only a moderate amount of design effort from the usability engineer (Figure 17) [41]. For example, a medium-fidelity prototype might be developed by the usability engineer using Microsoft PowerPoint™ that establishes the layout of major screen elements and major text labels. This is critical in that this allows the usability engineer to deliver designs in a timely fashion and be more flexible to make changes to the design based on changing requirements and implementation limitations. More specific details such as exact positioning of items on the screen and item colors can be provided as the developer implements the design through communications with the usability engineer. High-fidelity prototypes are developed by the usability engineer only for the most critical pieces of the user interface as determined by their relationship to the high-level goals for the system.

Claims. Some popular agile methods like extreme programming (XP) include the concept of test-driven development and automated testing [12]. Code tests can be aggregated and automatically run whenever the implementation for a particular feature is completed. This allows for very quick verification of new functionality using an increasingly complete test corpus, which helps agile teams maintain high velocity. Unfortunately, usability testing cannot be automated to such a degree, and user testing with real people is vital [55]; so, the usability engineer uses the concept of claims to help decide what aspects of the user interface to evaluate and why. Claims are feature descriptions with associated design tradeoffs that elaborate the user ramifications of different aspects of the interface [26, 111]. Claims are similar to stories or feature descriptions in that they are brief descriptions of

different parts of the system—however they are used to capture and guide design decisions rather than to aid in project planning and management. An example claim for the automated checkout system is presented in Figure 18:

Use of a voice-based auditory alert to notify the user when the next item can be scanned

- + Providing the user with the indication allows them to more quickly scan and bag items
- + An audio notification does not require the user to draw their attention to the display screen, allowing them to remain focused on the task at hand
- User may not hear the notification in a noisy environment
- User may become annoyed at the repeated notification

Figure 18. Example claim.

The usability engineer develops and associates claims for each set of design prototypes developed in an iteration. Only claims for features that directly relate to the high-level design goals are developed by the usability engineer. After a set of design prototypes are implemented by the developers, the usability engineer will then use those claims to plan the usability evaluation for the implemented functionality. To streamline effort required to maintain the CDR, the claims map that was originally developed as a part of the CDR has been simplified so that the relationships between claims are not explicitly maintained. Rather, the most critical link was found to be that between goals, design decisions, and the evaluations that validate those decisions (Figure 15).

5.3.4 User testing results

Just as agile teams rely on test driven development and acceptance testing to verify and validate the functionality of the system [69], usability engineers will conduct targeted usability testing throughout the development process to [111]:

- Verify that design decisions have resulted in a usable system
- Validate the user interface design to the team by showing that it meets high-level design goals
- Identify new and unforeseen usability problems
- Uncover new requirements

It is important to note that differences exist between how testing is done in agile development and how usability testing is done. In test driven development, developers can implement tests in code (aka unit tests) before they begin to implement the actual functionality. When the functional code is completed, the associated unit tests can be run to verify that the code is functioning as specified. In this way, the unit tests can act as the requirements for the code [12]. These unit tests can be aggregated and

automated such that anytime a developer changes the underlying functional code he can easily test to make sure that the changed code still meets those code requirements. The usability analogue in XSBD to unit tests are the claims linked to the high level project goals. The claims define what the usability engineer believes how specific UI features will benefit or potentially harm users. These claims are then verified through usability testing. Unlike in test-driven development, usability claims and tests cannot be aggregated and automated. Claims often include descriptions of effects which are subjective and difficult to quantify. In addition, if changes are made to a specific UI feature, it can have unintended and unpredictable effects on other areas of the UI. These differences require different usability testing strategies that differ from test-driven development. Usability testing—especially empirical end user testing—can be more closely compared to user acceptance testing. Acceptance testing is typically done in agile teams by the customer to verify that the system functions as he or she wants it to.

Usability testing will form the complete chain linking the high-level design goals to design decisions. The use of claims allows the usability engineer to streamline the evaluation process by focusing on only the most critical areas of the user interface. This is important as evaluations are planned and run within a single development iteration so results can be handed off to developers in the following iteration. Continually revisiting the high-level design goals with each test cycle will allow the usability engineer to run incremental tests while maintaining the overall vision of the user interface design. The tests are generally lightweight in nature and will rely on methods such as walkthrough evaluations with think-aloud protocols and expert reviews [93]. For example, the usability engineer may simply have the end user representative do a review of the user interface or have 3-5 potential end users at the client company use the part of the system being evaluated. These evaluations will often have to be run remotely as these end users may not be readily available to the usability engineer [52]. Essentially, the usability engineer will plan and run these evaluations within each iteration such that results can be processed and proposed changes provided to the team as soon as possible.

For each evaluation that is run, the usability engineer will review the high-level design goals and the set of claims developed for the part of the design being evaluated and use them to plan the evaluation. For example, the usability engineer may decide that the claim shown in (See Figure 19) is related to the first design goal and runs and evaluations the basic scanning process with four end users—making sure to collect timing and error data and asking questions at the end to get a sense of their frustration level with the system.

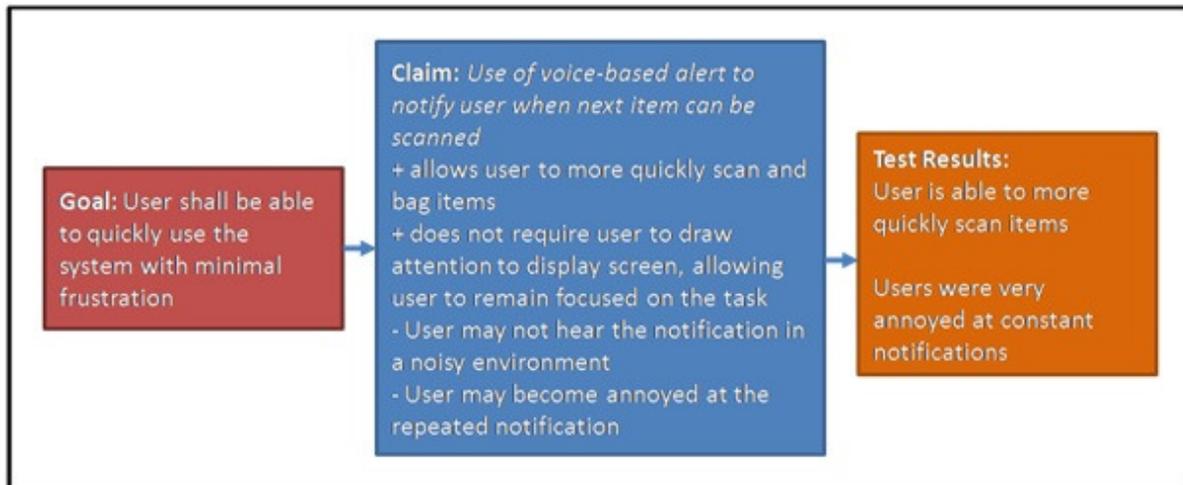


Figure 19. Relationship between design goal, design claim and testing result for the automated checkout system.

Based on the results of the evaluation, the usability engineer may propose a change to the design so that there is only a simple beep sound upon scanning the item: the voice-based notification will only occur 3 seconds after the scan if the system does not detect that it has been put in the bag. The usability engineer reasons that this will maintain efficiency levels while still helping users that are unfamiliar with the system. This suggested change to the design will be later presented to the rest of the XSBD team so it can be prioritized, estimated and put into the feature list.

Usability testing can also occur before a functioning version of the design is delivered by the agile developers. For example, the usability engineer can run a quick evaluation using an analytic evaluation technique such as a heuristic evaluation, or a cognitive walkthroughs as she is designing the UI for some part of the system [111]. This can allow the usability engineer to potentially find and fix usability problems before he or she hands off the design to the developers at the start of the next iteration so implementation of the design can begin. Analytic evaluations can also be used by the usability engineer to test the UI even if developers are unable to deliver functionality in a timely fashion.

In addition to these lightweight evaluations, more comprehensive evaluations are run by the usability engineer to provide more complete end-to-end testing of the user interface. These evaluations will typically involve a larger number of end users evaluating the system either at a usability lab or onsite at the client company. They will provide the usability engineer with a better idea of how usable the

system is and will verify the changes made to the design based on previous evaluations [111]. This type of evaluation is conducted less frequently as it is often time-consuming and may impact velocity.

5.4 Maintaining synchronicity through planning and processes

In the XSBD process there are separate but synchronized usability and development tracks. The usability engineer works one iteration ahead of the developers so designs can be delivered for the developers to start implementing at the start of each iteration. Through careful coordination of the development and usability tracks, the XSBD team can optimize its velocity while still developing a system that meets high-level design goals. This type of parallel development process is effective and is increasingly used to integrate usability and development efforts in agile usability teams [84, 97, 127]. This section details how this synchronized process works with XSBD, how the CDR is used to communicate design decisions and balance tradeoffs, and what kind of synchronization activities are used for the team to work together effectively. Note that this section gives only a basic overview of the planning and synchronization process used in XSBD. Variations to the synchronization process that are needed to address issues such as design and development delays or external constraints are described in Chapter 6.

5.4.1 XSBD process description

This section will detail how development and usability work together in the XSBD process. It will describe what each role does in each iteration, what artifacts—including the CDR, implementations, documents—are developed in each iteration, and who they are handed off to and why. Figure 20 gives a basic overview of the mainstream handoffs in the process. The usability engineer will develop a design for a part of the system (**D1**) which is then handed off to the developers to implement in the next iteration. The implemented design (**ID1**) is then given back to the usability engineer to evaluate (**E(ID1)**), which may result in changes to the original design (**D1'**), which is then handed back to the developers to implement in the next iteration (**ID1'**). This basic handoff procedure will be described in more detail below. Like other agile methodologies, the length of each iteration within the XSBD process can vary from 1-4 weeks depending on the size and contextual issues surrounding the system being developed. However, two-week iterations are commonly used [65]. The handoffs for a hypothetical release comprised of eight iterations is shown in Figure 21. A description of how this was implemented in practice will be described in Chapter 6.



Figure 20. The basic handoff structure between software and usability engineers.

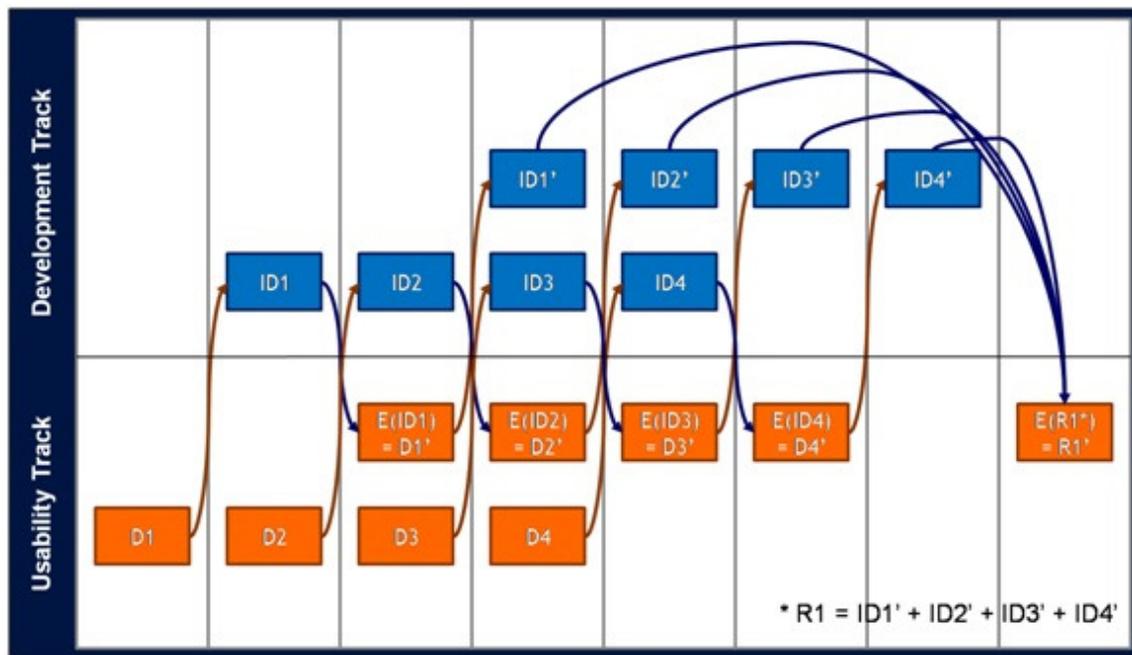


Figure 21. Parallel development and usability tracks for a complete release cycle.

Iteration 0:

The purpose of iteration 0 is to define the project vision and design goals for the project so that everyone on the XSBD team has an understanding of what system is being developed and why. This phase is commonly used in other agile usability methodologies for this purpose [97]. Iteration 0 is not meant to include feasibility studies or other market research. At this point in the project, the customer representative should have a general idea of what system he or she wants to be developed and why.

Near the start of the iteration, there will be a project kickoff meeting with all XSBD team members where the customer representative shares the project vision with the rest of the team. The customer representative should have a draft of the *vision document* prepared before the meeting. During this meeting, the team should define and prioritize the *design goals* for the project.

During iteration 0, the usability engineer will conduct an abbreviated requirements analysis process similar to that used in SBD, but designed to be completed within the iteration. This will include interviewing the customer representative, end user representative and other end users to gather information about current practice, contextual information, and information about the users. The output at the end of the iteration should be *user descriptions* and *scenarios* describing the major workflows. Additional requirements work can be done in subsequent iterations as needed.

In this iteration, the developers should take preparatory steps to beginning development. This will include things such as setting up the development environments, databases and any technologies that will be used to develop the system. This may involve the use of code spikes, where the developer starts coding in order to learn about a new technology or unfamiliar functionality implementation [12].

At the end of the iteration, the entire XSBD team will meet to generate the list of features that will be developed for the system. The customer representative is primarily charged with identifying the features and defining their priority, though he or she may get assistance from the usability engineer in the process. All material developed up to that point, including the *vision document, prioritized design goals, user descriptions and scenarios* will be available for reference. The usability engineer will estimate the amount of effort required to design each feature. This estimate should include both the time to design and to evaluate the feature. The unit of measure used to do the estimate will vary by the preferences of the team but could be in units like things like days, hours or points. Similarly, the developer will estimate the amount of time needed to design, implement and test the code for each feature using the same unit of measure. The prioritized and estimated list of features becomes the product backlog [115].

The project manager will use the estimates to develop a release plan. He or she will work with the customer representative and decide roughly what features will go into a release. The product being developed will be divided into releases with each release comprising some number of iterations. This high-level planning is meant to give the customer and the team an idea of what will be delivered within the release timeframe.

Iteration 1.

Iteration 1 is where the XSBD team starts designing and developing the system. Each iteration from this point on will start with a planning process where the team decides what will be done in the upcoming iteration. At the end of the iteration the team will review the work that was completed. The results of the review will then be used at the start of the next iteration's planning process.

Iteration 1 should begin with an iteration planning meeting where the developers and the usability engineer determine what features they will work on in the iteration. For the usability engineer, this will be to design the highest priority features that are on the product backlog. The number of features included will depend on the amount of time the usability engineer has available. For example, if the usability engineer has 64 hours (assuming 80% efficiency for a two week iteration), then he or she will take as many features as approximately fit into this time constraint. Because there are no designs for the developers to implement in this first iteration, they will work on parts of the system that do not require input from the usability engineer. For example, this might include developing the data model or building a database. Like the usability engineer, the developers will take on as much work as they think can be completed within the iteration.

Daily meetings will be held for the team members to meet to review progress. These 10-15 minute meetings should be run as specified in the Scrum process where each team member reviews what was done yesterday, what will be done today and what problems were encountered [115, 116]. The project manager will help team members resolve the issues as they come up and will manage and observe the project schedule. If work is progressing slower than planned, then development or design of features may be moved back to the backlog. If work is progressing faster than expected, additional features may be taken off the backlog.

At the end of the iteration, the XSBD team, including the customer representative and end user representative, will review the work completed in Iteration 1. For the usability engineer, this will be designs that the developers will implement in Iteration 2. This will include the *scenarios, design prototypes and claims* as represented by **D1** in Figure 21.

Iteration 2.

This is the first iteration where the usability engineer will hand off a design for the developers to implement. The project manager will work as before managing project progress and the daily meetings.

As in iteration 1, there will be a planning meeting where the developers and usability engineer determine what features will be worked on in the iteration. The usability engineers will take the next highest priority features and design them—into the mockups represented by **D2** in Figure 21. The usability engineer should set aside some amount of time (~10-20%) to work with the developers as they implement the design. This will involve clarifying or defining certain parts of the design or addressing problems related to implementation difficulties. The developers will work on implementing the designs from the first iteration (**D1**). At this point, the developers should be allocating time both for working on new features and for rework. Rework will include changes to the code that need to be implemented based on usability redesign, bug fixes and refactoring. The exact percentage of effort devoted to each type of rework should be determined jointly by the developer and project manager.

At the end of the iteration, the XSBD team will again meet to review the completed work. The usability engineer will present the designs that will be implemented in the next iteration (**D2**), and the developers will present the implemented designs (**ID1**). Upon acceptance by the customer representative, the team will proceed.

Iteration 3.

In iteration 3, back and forth handoffs between the developers and usability engineer will become clearer. The planning and review meetings will start and end the iteration as before.

The usability engineer will work on the designs (**D3**) that will be implemented in iteration 4. He or she will also conduct a usability evaluation on the implemented design delivered at the end of the last iteration (**E(ID1)**). The end result of this will be some changes to the design that will be added back into the product backlog (**D1'**). The usability engineer will work with the end user representative in planning and running the evaluation. The developers will implement the designs from the last iteration (**ID2**).

Iteration 4.

Iteration 4 will proceed as the previous iterations. It will be the first iteration where the developers begin to incorporate usability changes back into the product.

The usability engineer will work on designing the features for the next iteration (**D4**) and will evaluate the implementation delivered in the last iteration (**E(ID2)**) which will result in changes (**D2'**) that will be added back into the product backlog. The developer will work on implementing the designs delivered in the last iteration (**ID3**) and implementing the changes made based on the usability evaluations (**ID1'**). Note that which changes the developers actually implement will depend on the priority of those changes.

Subsequent Iterations.

Figure 21 depicts a release cycle of 8 iterations. The release is represented by **R1**. Near the end of the release, the usability engineer will run a more comprehensive usability evaluation (**E(R1')**) that will represent a complete end to end evaluation of the system as it exists with end users. This will also verify the usability changes made in reaction to the previous lightweight evaluations.

5.4.2 Synchronization points

Having parallel usability and software development tracks in the XSBD process has the benefit of maximizing the use of usability and development resources but introduces the potential problem of design drift—where the implemented system differs from what was initially designed [51]. In the XSBD process, this problem is mitigated through the use of enforced and opportunistic synchronization points.

Enforced synchronization points. These synchronization points consist of the regular iteration planning and review meetings that start and end each iteration. In the review meeting, the entire XSBD team gets a chance to review the work of each person. For the usability engineer, this means sharing the designs that the developers will be implementing in the subsequent iteration. This will give the developers a chance to review the design for feasibility and ask clarifying questions. The usability engineer will also use these meetings to share results from usability evaluations with the developers and share proposed changes to the design. The developers will use these meetings to present the implemented designs to the team which will allow the usability engineer to verify that the implementation matches the initial design. If discrepancies between the design and implementation

are found, they can be addressed in the subsequent iteration. The customer will also use this opportunity to validate that the designs match his needs and that the implemented system functions as initially requested. The customer representative or end user representative may identify new requirements or request changes based on what is seen. If so, the changes can be incorporated into the next iteration.

Opportunistic synchronization points. In practice, discrepancies between the design and the implementation can be found and addressed within an iteration through opportunistic synchronizations between XSBD team members. These may involve impromptu face-to-face meetings, emails or IM conversations. Opportunistic sync points between the usability engineer and developers are critical, as most of the delivered designs will be of medium-fidelity and there will be questions regarding the specifics of how the developed interface should look and function. In practice, these sorts of issues will arise regardless of the fidelity of the prototypes due to development constraints, changing requirements, and other issues encountered on a day-to-day basis. These sync points are also needed between the usability engineer and the customer representative and end user representative to resolve issues related to the design of the user interface.

5.5 *Balanced decision making*

One of the benefits of the CDR and how it is used in the XSBD process is that it can be used to help the team make sometimes difficult design decisions that balance different factors including usability, development effort and customer needs. It can also help resolve disagreements regarding the user interface design that might arise between team members.

5.5.1 *Balancing different concerns*

During the planning and review meetings, the XSBD team will have to make decisions about changes to the design proposed by the usability engineer based on usability evaluation results. The usability engineer will summarize results from usability evaluations and present changes to the design, with each change being assigned a relative priority level. For example, going back to the automated checkout system example introduced in Section 5.3, the usability engineer may present the following summary (See Figure 22):

| | |
|-----------------|--|
| Problem | Users are finding the voice notification to be disruptive and annoying. |
| Solution | Implement a simple beep sound when an item is scanned. Then if the item is not placed into a bag within 3 seconds, give the voice notification. + users will still be given audio cue for when an item is scanned |

| | |
|------------------------|---|
| | + experienced users will be less annoyed - new users of the system may not know what to do after scanning an item |
| Severity | High severity: 3 of the 4 participants stated that this was a serious issue. This problem directly relates to the two high-level design goals of efficiency and learnability. |
| Redesign effort | 2 hrs |

Figure 22. Summary of usability testing results and proposed solution.

The usability engineer will communicate the problem along with a proposed solution. He or she will also state the severity of the problem and support this assessment by directly linking it to one of the high-level goals defined in the CDR. The way that severity is communicated can vary depending on the preferences of the team, although a simple High, Medium, Low scale is typically sufficient. He or she will also give an estimate of how long it will take to do the redesign. The developer in turn will give an estimate of how long it will take to implement the revised feature(s). The customer may revise the priority of the feature or may need to add a priority to the feature if it is new. Based on this information, it will be placed on the product backlog and addressed in a future iteration.

Whether or not the feature is ultimately implemented depends on a number of issues:

- How important is this to the customer?
- How closely does it align with high-level design goals?
- How severe is the usability problem?
- How much effort will be required to design and implement the change.

For example, if the change is directly related to the highest priority design goal and will take a minimal amount of effort to implement, then it will most likely be implemented in the final design. However, if the problem is of moderate severity, does not relate to the highest priority goal, and will take a significant amount of development effort, than it may not be implemented. This process is similar to how severity ratings are used in heuristic evaluations to prioritize usability problems [94]. The team will have to decide collectively based on the information provided by the CDR and from the team members themselves. It is not sufficient to simply rely on the customer for guidance in all cases, such as those where the problem relates to a complex or obtuse usability issue.

A similar process will occur for other types of changes including new requirements or design changes required due to implementation or technology-related limitations. Using the CDR will make the interaction between the different issues more salient and provide guidance for future actions.

5.5.2 Claims to compare design features and resolve disagreements

Since the user interface is the most visible portion of system under development, all team members including developers and the customer representative will give feedback related to the user interface design. While this is useful and valuable in many instances, such as when brainstorming ideas about the interface, this can result in tensions between team members. To mitigate this issue, claims can be used to resolve these kinds of disagreements. If another team member proposes a change to a system feature as designed by the usability engineer, the usability engineer can record the suggestion in the form of a claim. This is a simple way to record the alternate design and its tradeoffs. Although development will typically proceed with the original design, documented alternate designs may be considered if implemented features do not satisfy usability standards at test time.

5.6 Chapter summary

This chapter provided an overview of the XSBD approach in its most advanced form. It described how the XSBD process was developed so that key usability practices from SBD could function within an agile, incremental development framework. It defined what the key roles within the XSBD approach were and outlined responsibilities of each. It then introduced the primary design representation used in XSBD—the central design record (CDR)—describing what it was used for and what its major components were. Specifically, it focused on how the CDR allows the usability engineer to work incrementally while still maintaining a high-level overview of the UI design by directly linking high-level goals to design features that are validated through usability evaluations. Next, this chapter described how usability and development processes operate in parallel and how the two process tracks remain in sync through coordinated planning and communication mechanisms. Finally, it described how usability and development concerns are balanced using elements of the CDR.

The next chapter discusses how the XSBD approach described here was iteratively developed and substantiated using an action research methodology in development efforts at a software company.

6 Substantiating XSBD in practice

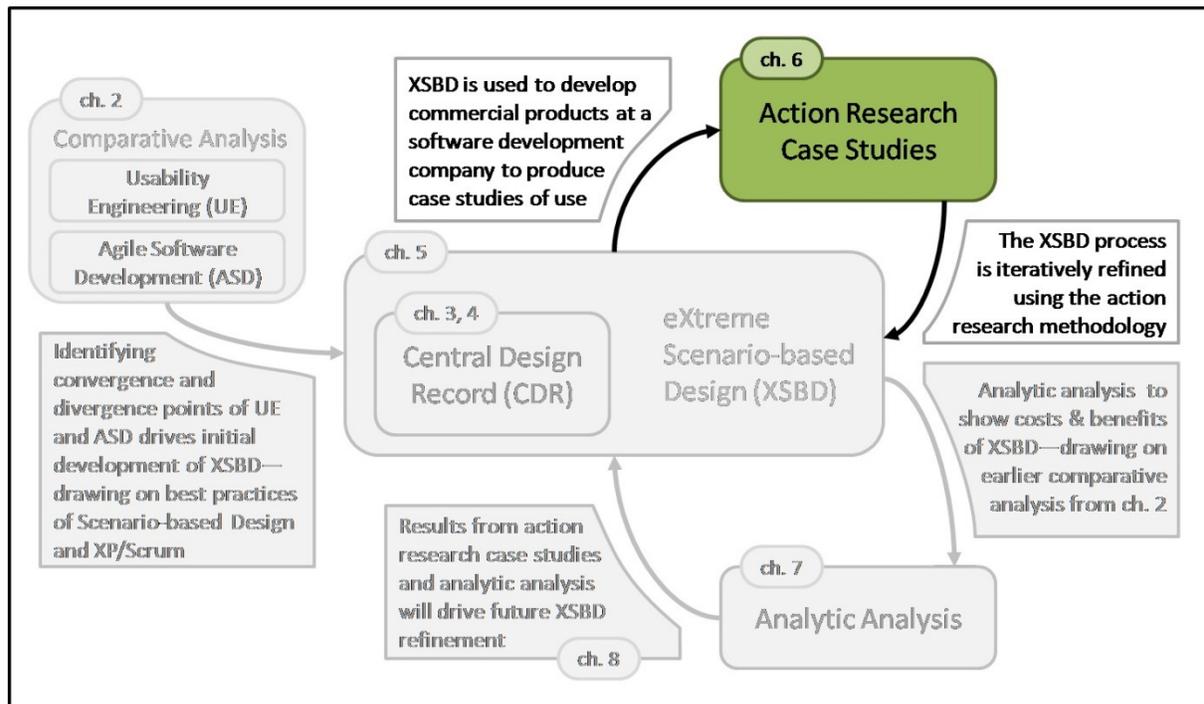


Figure 23. Overview of solution approach. Chapter 6 will summarize how the XSBD process was substantiated and refined using an action research case study approach.

This chapter will present the approach used to further develop and substantiate the utility of the XSBD process (See Figure 23). Portions of the work in this chapter have been published in my Agile 2009 conference paper: *Examining the Foundations of Agile Usability with eXtreme Scenario-based design* and used to develop an agile usability tutorial presented at the ACHI '09 conference: *Working in Harmony: Integrating the efforts of usability engineers and agile software developers* [65, 66]. It will address the last three issues presented in *Section 1.1 Challenges facing an integrated approach*.

2. **Balance of power.** In developing a specific instantiation of a combined approach, how should the approach be balanced between usability and agile methods? For instance, an agile-centric approach may involve providing some level of usability-training to agile developers while a usability-centric approach may have usability engineers complete UI designs before agile developers begin their work. The integration approach will have an impact on the long-term viability and acceptance of an integrated process in practice.
3. **Checks and balances.** How can appropriate tradeoffs between the agile and usability practices in the integrated approach be made? Any integrated agile usability approach will have tradeoffs that may, for example, sacrifice development velocity for validation of the usability of a user interface. The integrated approach should help agile usability teams make these tradeoffs given the specific needs of their development project.

4. **Synchronicity.** How can the agile development and usability engineering sides of a project stay synchronized throughout the development effort? This is critical as design drift between the interface design and its implementation can render the usability work irrelevant and result in a system that has poor usability. This problem is exacerbated by the speed at which agile development projects proceed. Addressing this issue depends on proper development and use of shared design representations, communication practices and process planning structures within the team.

It will show why the XSBD approach was developed to be agile-centric rather than usability-centric and how that helped adoption of the approach in practice. It will also show how balance between agile and usability concerns is maintained through continuous consideration of design concerns centered on the CDR. Finally, it will show how simultaneous usability and development efforts are synchronized through structured communication and collaboration practices.

The substantiation approach centers on the action research methodology. In this approach, the XSBD process is actively used and improved through its use in developing software systems at a software development company while simultaneously making changes to the approach based on continuous feedback from the project participants. The approach was applied in practice through a partnership with Meridium, Inc., a software development firm based in Roanoke VA. Meridium was in the process of transitioning to agile development methodologies while simultaneously increasing their focus on usability methods—making them an ideal candidate for the XSBD process. The XSBD process was used to develop two software systems at Meridium over the course of one year. This chapter will present an analysis of the results from those case studies—describing how the process was adapted based on feedback from the developers and on specific project conditions. It will highlight the strengths and shortcomings of the approach and summarize variations of the approach that are made based on specific project conditions.

This chapter will first present an overview of the action research and why it was used to substantiate the XSBD approach. Second, it will summarize the partnership with Meridium, Inc. and give an overview of the development projects XSBD was used on. Third, it will provide a detailed summary and analysis of how XSBD was used and refined in practice through its use in developing the software projects at Meridium. Finally, it will summarize the benefits and limitations of the XSBD approach based on the results of the action research case studies.

6.1 Substantiation approach

This section will summarize the approach used to evaluate and improve the XSBD process.

6.1.1 Action research

Action research is grounded in the concept that complex social processes are best studied by continuously introducing changes and observing effects on the phenomena of study [124]. For this reason, it is increasingly being used to study information system methodologies which typically involve groups of people collaborating on some project [7, 9, 39, 40]. This also makes action research ideal for both evaluating the effectiveness of XSBD and iteratively improving it. By applying XSBD in practice using this approach, I am simultaneously assisting in practical problem solving while expanding scientific knowledge of agile usability practices, thus helping to close the gap between theory and practice [23, 70, 101, 135].

Action research is divided into a series of five stages forming a complete cycle that is continuously iterated upon, as shown in Figure 24[124]. Problems are first identified and *diagnosed*. In the *action planning* stage, researchers and participants identify what actions will mitigate those problems before moving on to the *action taking* stage where those actions are implemented. In the *evaluating* stage, researchers review the results of those actions and how well they addressed the specified problems. In the *specifying learning* stage, researchers develop and build on theories based on the results of this and past cycles.



Figure 24. Stages of action research [124]

During each case study, I followed the five stages of action research to iterative improve the XSBD process. I observed and collected data as the project team members worked to develop their respective systems. This data included critical incidents, design artifacts and results from retrospectives and

interviews (See Section 6.1.2). During each project, specific problems were identified (*diagnosing*) which were then addressed through specific actions defined by myself and the team (*action planning, action taking*). I then reviewed the result of each action (*evaluating*) to see how well each action addressed an issue. Thus, I was actively collecting and analyzing the data from each project as they were going on. Certain high-level patterns emerged which formed the basis for my later post-project analysis. After each project ended, I reviewed the data collected from each case study, identifying high-level patterns and results which are summarized in Sections 6.3 and 6.4 (*specifying learning*).

6.1.2 Data collection

Proceeding through the stages of action research, I collected data using a number of different methods throughout the development projects. I was present at Meridium throughout both of the development projects the XSBD process was used in and documented *critical incidents* as they are observed [44]. Critical incidents are observations of behavior that have significance to the phenomena being studied—in this case the XSBD process. Critical incidents contained the following information:

- The person or persons involved
- The date of the incident
- A brief description of the incident
- An explanation of how this negatively or positively affected the team.

Observing and documenting issues as they came up helped in identifying problems to address—the first stage of the action research process—and also helped in seeing what effect they had on team performance.

Critical incidents were primarily documented during the various team meetings including daily meetings, weekly team meetings and iteration review meetings where team members communicated progress and any blocking issues. I also tracked incidents by tracking email exchanges through each team's email list and by periodically observing opportunistic interactions between team members.

Team members were also asked to self-report critical incidents by using an online submission form I set up for each team. This was used because I was unable to observe team members at all times during the development projects. Hartson et. al found that self-reported incidents were comparable in quality to incidents observed by usability experts [52]. This was also useful for me in that it captured different perspectives when multiple team members reported on the same incident. A copy of the online critical incident form used to self-report incidents is shown in Figure 25.

Critical Incident Form

Meridium-VT NSF STTR Agile Usability Project

What is your role in the project?

Manager Usability Developer Tester Other:

Date of incident:

Briefly describe the event that either positively or negatively affected team communication and/or collaboration. Please provide any necessary background information related to the event.

As I was implementing the layouts for the search screen based on the UI prototype I did not understand why we were using a grid layout instead of tabbed pages. I talked to the usability engineer, and she explained that she had conducted an evaluation with the users and found the grid layout to be a more efficient way to input a lot of data.

Why do you believe this positively or negatively affected communication and/or collaboration?

I believe this positively affected team communication since the usability engineer was on hand to immediately answer my question. Previously, the prototype screens would be done weeks or months beforehand and the UI designer would not be easily accessible.

How would you rate the magnitude of the effect (positive or negative) that this incident had with respect to overall development progress?

- 1 - low effect
- 2 - medium effect
- 3 - high effect

Figure 25. Sample online critical incident form.

I regularly collected work products including design mockups, project schedules and meeting notes. This helped me to keep track of project progress and to make connections against the critical incidents collected throughout the projects.

I also conducted a retrospective at or near the end of the release cycle for each project. Retrospectives are often used in software development teams to reflect on team processes, and are conducted periodically in agile teams to ensure that the team is operating effectively and to improve processes as a project is ongoing. After reviewing some of the data collected at the end of the project, I also conducted individual post-project interviews from core project team members to collect additional subject feedback about the project.

A listing of the collected data for each case study is available in Appendix A, Appendix B, and Appendix C.

6.2 Collaboration with Meridium

To evaluate the use of XSBD practice, I partnered with Meridium, Inc., an international software and services firm based out of Roanoke, VA. Meridium specializes in developing asset performance management solutions for process, power, mining and discrete manufacturing industries [82]. These systems are used to increase the reliability of and improve the performance of production assets while optimizing maintenance costs, enabling regulatory compliance and increasing safety.

Meridium proved to be an ideal candidate for the XSBD process for two reasons:

- 1) They were instituting a company-wide effort to transition from a water-fall based development process to an agile development process.
- 2) They were interested in adopting usability processes to maintain competitive advantage over their competitors. This was the initial reason that they started working with me.

This collaboration was supported through a National Science Foundation Small Business Technology Transfer (STTR) Phase I Grant entitled “Integrating Scenario-based Usability Engineering and Agile Software Development Practices” (IIP #0740827) [88]. I wrote this grant with Scott McCrickard—my advisor at Virginia Tech—and Hari Pulijal—vice president of development at Meridium—specifically to support the evaluation of the XSBD process in practice. The XSBD project was used in two development efforts at Meridium over the course of one year as supported by the STTR grant.

Project 1: Web portal application

The first project that used the XSBD process was a web portal application that allows customers to have online access to some of the features available in the main Meridium desktop application, Asset Performance Manager. This project was chosen because it required significant UI design and development. In addition, it represented a relatively small development effort that could be completed within the timeframe of the STTR Phase I grant. Core development effort for this project ran from February into May 2008.

Project 2: Touch screen application

The second project that used the XSBD process was a touch screen application to improve a paper-based work order system by having mechanics directly enter work order information into the company database through the touch screen. This project was chosen because touch screen user interfaces require usability expertise to design effectively and was a relatively small self-contained project that had minimal dependencies to other development efforts at Meridium. This development effort ran from June to October 2008.

6.3 First case study: Web Components

Meridium's Asset Performance Management (APM) software forms the cornerstone of Meridium's software suite. It is a desktop application that "enables a set of work processes to maximize physical asset performance, mitigate risk, and optimize cost in a business enterprise" [82] using a variety of sub-module applications that allow companies to collect and analyze large amounts of data and then develop strategies for optimizing the practices of using and maintaining large assets. As such, many of the modules require specialized knowledge, expertise and training to use correctly. Web Components was the name given to the development effort to provide an easily accessible interface to the data stored by the APM product, allowing casual users to be able to see the data without the overhead to learning to use the main desktop product.

The development team was the first at Meridium to adopt the XSBD process. This project was unique in that the team was distributed with many of the developers were offsite, while the customer-facing team and usability engineer were located at Meridium's headquarters in Roanoke, VA.

A total of 90 critical incidents were collected of which 14 were self-reported by the Web Components team members. In addition, a retrospective was held at the end of the fourth iteration. A complete listing of these results is available in Appendix A, Appendix B, and Appendix C

The following section will give a description of the project personnel, the system being developed by the team and then describe findings from the case study. Results from the case study will be presented in narrative form with major issues and actions taken highlighted and described in terms of the stages of action research: What issues are identified (*diagnosing*), what actions were taken to address them (*action planning and action taking*), and how well those actions addressed the issue (*evaluating*). The end of this section will summarize lessons learned and major findings (*specifying learning*).

6.3.1 Project personnel

The project personnel consisted primarily of experienced Meridium employees. This project was unique in that the team was distributed with most of the development and quality assurance (QA) team being located at a different site while the customer-centric team consisting of the project manager, customer representative and usability engineer was based out of Meridium's main office in Roanoke, VA. The team consisted of seven development/QA members (four of whom were offsite), one usability engineer, one product lead acting as the customer representative and end user representative, one development lead, one project lead and one documentation person. The usability engineer was recruited from Virginia Tech and had some previous experience in interface design and development from graduate-level courses in Human-Computer Interaction and Usability Engineering. The development lead helped to coordinate the work of the offsite team members and acted as a liaison to the locally-based team members.

The offsite team members had extensive experience with agile methods. They primarily relied on Scrum to manage their development efforts. The onsite team had limited experience with agile methods as Meridium had only recently started transitioning to agile development methodologies.

As is typical at Meridium, all team members were not exclusively working on the Web Components project and spent on average 50% of their time on it. The usability engineer was a Human-Computer Interaction graduated student recruited from Virginia Tech and worked 20 hours/week on the project.

6.3.2 Project description

Project visioning for the Web Components project was largely complete by the time the team was put together in early 2008. The product manager had completed a vision document outlining the business need for the product, how the product will be positioned and a high-level overview of the requirements of the system. The Web Components system was described as a system to allow casual users to access information stored in Meridium's core Asset Performance Management product through a web

browser. The intended users are upper management personnel who have limited experience using their core product and need an easy way to view data and reports output from it.

By the time the usability engineer joined the team in the middle of February, most of the offsite development and QA personnel were in place and ready to start development. Development was to be divided into iterations with weekly iteration review meetings where the team would review the current implementation and plan the subsequent one. In daily standup meetings, the team would share with each other progress made, identify impediments and discuss ways to address them. This general framework was put together by the offsite development lead. The daily meetings were scheduled so they were at the beginning of the day for those onsite at Meridium and were thus at the end of the day for the offsite developers.

Although the offsite team members were experienced with agile methods—Scrum in particular—but they did not have previous experience working with a usability engineer. This led to several collaboration problems between the usability engineer and the offsite developers.

Issue 1. Limited understanding of the value of having a usability engineer on the team.

Diagnosis: There was a general misunderstanding of what the role of the usability engineer was on the team. There was an initial phone conference meeting between the offsite developers and the usability engineer at Meridium to discuss the development process to be used and what the usability engineers' role would be. The offsite developers only expected the usability engineer to provide mockups for them to develop on and had no understanding of usability tasks including requirements analysis activities and user testing processes. A similar problem existed for other Meridium team members including the product manager and project lead as most employees at Meridium had limited experience working with a usability engineer.

Action: To address this issue, the team held an official 'kick-off' meeting where all team members were assembled to give an overview of the project, the development process and to describe what the usability engineer's job would entail.

Evaluation: The initial kick-off was found to be of limited benefit to the rest of the team in highlighting what the usability engineer did and what her value to the team would be. The offsite developers had already started initial development work before the entire team was assembled. Other team members including the documentation person, and several developers were initially committed to

other projects so the official ‘kick-off’ was delayed by two weeks. This had a negative effect in that the developers started implementing before understanding how they would work with the usability engineer. The kick-off meeting necessarily focused more on introducing the Web Components project to the other team members. Even introducing the project proved to be a challenge as new team members had trouble understanding what the system would do and why. The product manager noted that:

“Some team members were lost and the team does not have a common level of understanding.”

We later found that the best way to get other team members to understand the role of the usability engineer was for the usability engineer to present work products such as mockups and user testing results to the team and show them what was being done and why (See Section 6.4.2: Issue B).

As some of the offsite developers were available to work on the project before the whole team was assembled, the project lead had them use the first iteration to begin development on some of the high priority items by working on development spikes related to the underlying architecture of the application. Development spikes are used in agile teams to explore solutions to difficult design or technical problems by coding them up and evaluating them. This led to two problems, which are described below. First, they did their UI design/prototyping work without considering high-level design goals or end user needs in general. Second, these spikes involved UI design work, which the offsite developers did without consulting the usability engineer.

Issue 2. Team did not have common understanding of high-level design goals for the system at project inception.

Diagnosis: The team did not define or agree upon usability goals and metrics at the inception of the project. In addition, usability engineer had difficulty getting the information needed to define and share those goals with other team members before development work started on the project. The product manager, who was acting as the customer and end user contact, was unavailable at the start of the project because of commitments to other development efforts. This lack of common understanding of the usability goals of the system proved to be problematic early on as the offsite developers ended up doing much of the early design and implementation work without consulting the usability engineer and without consideration of those goals. In addition, suggested UI changes from

other team members at times conflicted with those high-level goals and decisions were made without consideration of the tradeoffs of different design options and their relationship to those goals.

Action: After meeting with the product manager to question him about user characteristics and product features, the usability engineer did modify the vision document to include usability goals and metrics—one of the most important pieces of the CDR—and presented them to the team at the kick-off meeting. The goals were defined as shown in Figure 26:

High level usability goals:

The system will be easy for novices to use, will be efficient, and will have high learnability. The system will provide a familiar feel to the APM, which will allow expert users to discover features easily and be easy enough for novices to learn without extensive training.

Figure 26. High level goals for Web Components project.

The most important requirement of the product is that it be quick and easy for novice users—those who have limited previous experience with Meridium’s core product—to use, while also being easy for them to learn. The user interface should, where possible, also follow existing interaction and visual design conventions from Meridium’s core APM product so experts will also be able to use the system easily. These general goals were shared and were designed to be understood by all team members. The usability engineer also defined specific usability metrics related to these goals to be measured in usability evaluations. They were defined as shown in Figure 27:

Usability metrics:

The system will:

- Take novice users an hour or less to discover and learn the functionalities available.
- Enable users to customize the web page to their individual needs in less than half an hour.
- Enable users to view a report using three clicks or less.
- Enable users to create a new record using three clicks or less.
- Enable users to find an existing record using the search functionality.
- Enable users to expand/collapse child record easily.
- Move a record from one parent to another using three clicks or less.
- Enable users to use the search functionality to find a desired item in 30 seconds or less.
- Enable users to use the keyword search to find a desired item in two or less attempts.

Figure 27. Metrics for Web Components project.

Evaluation: Although the usability engineer defined and shared these goals and metrics with the rest of the team at a relatively early point in the project, they were not accepted or used as guidelines by the rest of the team consistently at the start of the project. This was because the other team members, such as developers and documenters, had their own concerns and issues that they focused on. For example, documentation tended to focus more on keeping the Web Components product consistent with the core Meridium APM product rather than place more emphasis on ease of use as defined by the high-level usability goals. The usability engineer continuously acted as the ‘user advocate’, providing rationale for why certain design decisions should be made which were then considered by the team as a whole in making decisions as to how to prioritize a design change within a release. By the fifth iteration, the team as a whole appeared to have a better understanding of what the design goals of the system were and how they were prioritized.

Issue 3. Development started work implementing the user interface before the usability engineer could finish design work.

Diagnosis: The offsite developers began implementing features during the first iteration before the usability engineer was able to complete an initial requirements analysis process and begin any design work. This work was done without consideration of high-level design goals (which had not been defined yet), and was largely an attempt to copy the existing look and feel of Meridium’s APM product. In addition, the usability engineer noted that:

“This has negatively affected collaboration as I feel left out of the loop. The development team and the lead being in another country does not help.”

This also had the negative effect of marginalizing the role of the usability engineer within the project—making it difficult for her to demonstrate the value of usability-related work to the team.

Action: At the kick-off meeting the product manager and usability engineer briefly described the role of the usability engineer, explaining that the usability engineer will work one iteration ahead of the developers, providing designs for the developers at the start of each iteration for them to implement. In addition, the product manager—who was acting as the customer and end-user contact—took on the role of developing some of the mockups which were later reviewed by the usability engineer. Finally, where possible, the developers were assigned work that was not dependent on the UI designs or were assigned to do rework and bug-fixes. For example, after the kick-off meeting, user-interface design

work was moved to the product backlog so it could be worked on in later iterations. This provided the usability engineer with more time to work on designs for the next iteration.

Evaluation: By the fifth iteration of the Web Components project, the usability engineer was able to catch up with the developers and begin working ahead of development. However, there were persistent instances throughout the project where developers made usability design decisions without consulting the usability engineer. This was due to the fact that the developers and usability engineer were unable to work and communicate synchronously and because development velocity remained very high throughout the project—making it difficult for the usability engineer to keep up (See Issue 5).

One way to address the problem of having developers working faster than the usability engineer was to find ways to use the CDR to streamline the usability engineering process. The goal was to direct the usability engineer to work on only the most important areas of the user interface—those that will provide the most value to the end users. In addition, we wanted to minimize the amount of work the usability engineer had to do in updating and maintaining the CDR so that she could focus her efforts on designing and evaluating the system at hand while still drawing key benefits from it.

Issue 4. Need to streamline the usability engineering design, evaluation and documentation efforts to provide optimal value to the team without limiting development velocity.

Diagnosis: The team as a whole did not want to slow development velocity to wait for the usability engineer to conduct extensive requirements analysis and design activities. The team wanted to continuously work on and deliver software at the end of each iteration. Thus, if the usability engineer was unable to work in the given time constraint, the developers would work on without heading from the usability engineer.

Figure 28a shows the typical handoffs that occur between the usability and development tracks on an XSBD team. The usability engineer will work on a design (D1) in the first iteration, which will then be handed off to the developers to implement in the second iteration. The implemented design (ID1) will be evaluated by the usability engineer in the third iteration, resulting in design fixes (D1') which will be folded into the product backlog to be implemented in subsequent iterations (ID1'). Figure 28b shows what happened in the Web Components project. The usability engineer is unable to complete the first design (D1) before the developers begin implementation. The development team came up

with their own design (DA) which they implemented (IDA). Similarly, feedback from the usability engineer (DA') was not as useful to the developers as it was delivered late, after the development track has worked on and perhaps made significant changes to the system. Since the usability engineer could not keep up with development, her work tended to get ignored and she was effectively cut out of the development loop.

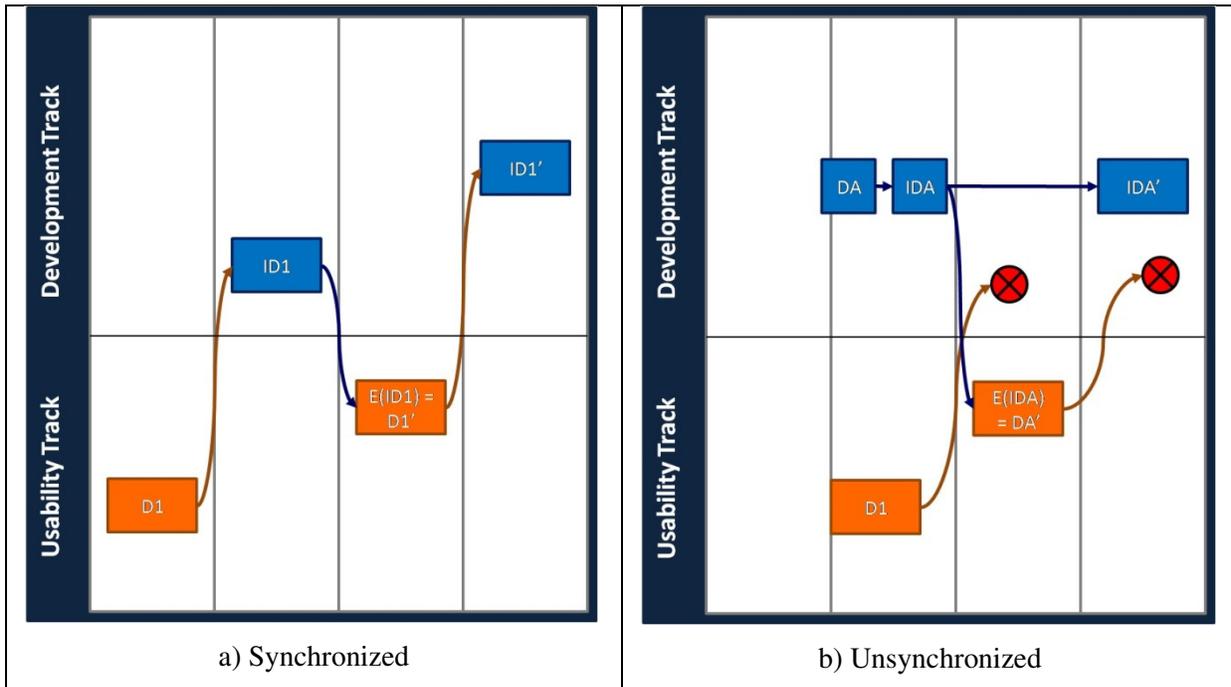


Figure 28. Image A shows that development and usability are synchronized. Image B shows usability work being delivered late and being disregarded by development as a result.

Action: Previously, the claims map was intended to be used to link usability design goals to specific features and allow the usability engineer to maintain a complete high-level view of the user interface design [64]. This was because the claims map could be developed incrementally as specific features were implemented in each iteration. One criticism of this approach was that the cost of developing and maintaining a claims map would be relatively high and would prevent the usability engineer from focusing on immediate design and evaluation concerns. This type of criticism is common in approaches that rely on design rationale both in the HCI and software engineering domains [6, 20]. In this effort, we decided not to require the usability engineer to maintain a claims map for two reasons. First, this was a relatively small-scale project such that the usability engineer would be able to maintain a good understanding of the overall design without explicitly documenting the claims relationships. Second, in keeping with the agile principle of valuing working software over comprehensive documentation [13], we wanted to minimize usability engineering work that did not directly lead to a functional system. Instead, the usability engineer developed and maintained a

lightweight list of critical design claims as she worked on the user interface design (See Figure 29). These claims were related to high-risk features and were marked as ‘resolved’ when they were validated by the usability through evaluations or addressed in some other way by a collective decision from the team.

| |
|--|
| <p>Using right click functionalities for Record Explorer tasks (Resolved)</p> <ul style="list-style-type: none">+ Reduces clutter on the left task bar+ Left task bar will always have two sets of actions+ Similar to the rest of the web component- Back to right click functionality <p>The team decided to use right click for some of the tasks.</p> <p>Removing Hide Record Explorer Task Bar (Resolved)</p> <ul style="list-style-type: none">+ Saves space+ Does not provide visual or cognitive clues of its use, might confuse users+ Not consistent with other items that also allow expand/collapse |
| <ul style="list-style-type: none">- Need to provide a method to expand/collapse the Record Explorer <p>This issue was resolved as the usability team conveyed the importance of consistence and the importance of providing a good visual cue for the users.</p> |

Figure 29. Key design claims maintained by the usability engineer.

Evaluation: The list of claims helped the usability engineer plan evaluations by defining what areas and features needed to be tested and was also relatively easy for her to maintain. The CDR is meant to help communicate design rationale to other team members to make sure they understand what work the usability engineer is doing and why. As it was used, team members—especially the offsite developers that the usability had the least contact with—did not seem to find much value in her work (See Issue 5). The relationship between the design goals, the designs and the usability evaluations was not clearly demonstrated to or shared with the team.

The first iteration was extended to three weeks because the team at the time had not been completely assembled. Quality assurance personnel and documentation were only able to join the team near the end of the first iteration. By the third iteration, the team settled on a one week iteration cycle where work would be delivered at the end of each week and then reviewed near the beginning of the following week. However, even as the off-shore developers were able to start working on features and

delivering work on a regular basis, there were continuing problems with how well they were communicating and collaborating with both the usability engineer and other team members who were onsite at Meridium. For example, during the second and third iterations, it became clear that the team was having problems integrating quality assurance testing into the development process.

Issue 5. Offsite developers are making design changes to the user interface without consulting the usability engineer.

Diagnosis: Even after the usability engineer began delivering designs to the developers to implement on a regular basis, there were still instances where the developers made changes to the user interface without first communicating with the usability engineer. This had a negative effect on both future usability work efforts and on general testing efforts. One onsite QA person noted that:

“Development deliveries are being made for Web Components, but not based on usability Mock ups... This has a negative impact because it causes lots of confusion and dev and testing rework.”

QA used the designs delivered by the usability engineer to begin test planning activities. If implementation did not match the design then she had to redo the test planning work. In addition, the usability engineer’s designs for the next iteration had to be reworked as they may have been based on her previous design. Also, usability test planning had to be reworked to match what was implemented. These changes were typically made for several reasons. For example, the delivered design may be incomplete and the developers made their best guess for what to do in a scenario not defined in the design. In addition, the delivered design may not be easily implementable and they may have altered the designs accordingly.

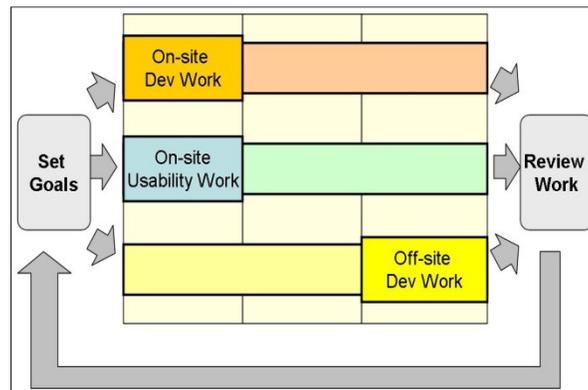


Figure 30. The workday split into three eight hour slices. Onsite and offsite work proceeds asynchronously so issues cannot be addressed until at least the end of a 24-hour cycle.

Part of the problem was that the onsite and offsite team members could not work synchronously as they were distributed (See Figure 30). If any issues arose, they could not be communicated until the next daily meeting. Another part of the problem was that the offsite developers were accustomed to designing the software they implemented and typically choose to work through design issues themselves.

Action: The team clearly defined the mechanisms through which the usability engineer would deliver designs and how the team would communicate design issues as they arise. For example, some actions the team took were documented in the retrospective held after the fourth iteration:

***Issue:** How do we disseminate information about UI design to the entire team? <Usability Engineer> and <Product Manager> will hold an extended UI meeting on Mondays but the entire team is unable to attend.*

Proposed resolution: After the UI meeting on Monday, <Usability Engineer> and <Product Manager> will touch base with <Development Lead> to review the feasibility of implementing proposed UI changes. Then <Usability Engineer> will upload the UI changes to the portal along with a list of changes with related design rationale. <Product Manager> will then present these changes at the daily meeting on Tuesdays.

***Issue:** How to keep track of important updates to the Portal so action items and open issues are addressed at some point?*

Proposed resolution: Once a week (at the feature team meeting?), the team lead needs to present any messages & action items that have been posted to the Portal so they can be reviewed and resolved. Also, if anything important is updated/uploaded to the Portal, then that person should send an email out to the Web Components team so they are aware of the change.

In addition, if the usability engineer saw any major discrepancies between the design and the implementation, or if the team has some specific concerns about a design feature after seeing it in action in a demo, she would document them in the form of claims which were either later tested by the usability engineer or otherwise resolved through the consensus opinion of the team.

Evaluation: More clearly defining the communication and information sharing mechanisms helped to address many of the communication problems between the offsite developers, the usability engineer

and the team in general. The problem of the offsite developers making design changes without consulting the usability engineer subsided as they grew accustomed to working with her and the new way of communicating. Claims were useful in that they helped the usability engineer document design inconsistencies and disagreements and determine how they should be addressed. Finally, the online team portal proved to be very important in that it served both as a central information repository and communication mechanism that mitigated some of the communication problems of having a distributed team.

Issue 6. Team did not have a clearly defined way to integrate QA testing into the XSBD process.

Diagnosis: The QA personnel at Meridium—who are in charge of doing functional and regression testing—were largely unfamiliar with agile processes and had trouble determining what to test at the end of each iteration. This was especially problematic as the offsite developers were working at such a high velocity and were not working synchronously with the QA person at Meridium due to the time difference. One of the QA personnel at Meridium noted that:

“there are no work tickets written to describe the delivery...Dev is working on completing Iteration 2 and QA does not yet know what to test for iteration 1.”

QA was used to having developers deliver work items to them using the online work management system Meridium was using. However, this team was relying on a product backlog that was being managed using Excel spreadsheets. In addition, many small fixes and changes were being communicated informally by team members at the daily meetings and iteration review meetings and were not being captured in the work management system. The QA person at Meridium noted that if these changes are not recorded anywhere then QA will not know what to test against. The project lead commented that many of these issues were trivial and inputting them into the work management system would take up to ten times longer than actually fixing them.

Action: The team compromised by deciding that within an iteration, changes and fixes could be communicated informally and did not have to be documented. However, once work for an iteration was delivered, the developers should input work items into the work management system which QA could then use to determine which items to test. This action was documented in the retrospective held at the end of the fourth iteration:

Issue: *How are small UI changes communicated to developers so they are actually implemented by the developers?*

Proposed resolution: If the proposed changes are a part of the current iteration (i.e.

they have not been delivered yet), then they can be communicated informally to developers. If the changes relate to features that have already been delivered, then they need to be written up as tickets...

Evaluation: This compromise allowed QA to begin testing the system consistently as they could have a better understanding of what features have been delivered and are ready to test. Within an iteration, the team was able to work quickly and informally to fix changes and problems as they were identified. For example, oftentimes a team member would identify a problem with the current system, communicate it to one of the offsite developers at the daily meeting, and it would be fixed by the following day. However, changes for delivered work were more time consuming as team members had to work through the work management system and any changes had to then be retested by QA. In addition, the QA person at Meridium found it hard to do longer-term planning which affected how the company planned and maintained dates for major releases. For example, if work was moved from an iteration to the backlog, the team did not have a well defined way of determining what impact that would have on the release schedule. Although long term planning is not a focus of agile methods, such plans matter in enterprise-level organizations that need to plan multiple releases and determine how to allocate a large number of resources [17].

The question of how best to integrate usability and QA into the development process was not the only factors that affected development velocity and team collaboration. As the Web Components system had to interoperate with the underlying database system of the core APM product, the team was dependent on some changes/fixes that the team responsible for that part of the system (the Core team), had to implement.

Issue 7. External dependency to another project team limited development progress

Diagnosis: In the fourth iteration, the team determined that some backend changes needed to be made to make the Web Components search functionality work as it was designed by the usability engineer. These backend changes could not be made by the developers on the team because they dealt with functionality that was being developed by the Core team. As the Core team was dealing with change requests from multiple teams within Meridium, the requested changes were not immediately implemented. As a result, the implementation of the search functionality slipped into the seventh iteration that proved problematic for the usability engineer, who was unable to run a usability evaluation on that part of the system.

Action: The developer in charge of implementing the search functionality, who was the one developer working on site at Meridium, was charged with being the contact point between the Web Components team and the Core team. Because this developer worked in the same place as the Core team members, she was able to keep in regular contact with them and communicate their progress on the needed functionality to the rest of the Web Components team members.

Evaluation: Having someone on the Web Components team act as a sort of ‘envoy’ to the other team helped keep the team informed about what progress was being made in implementing the needed changes and what issues were keeping them from completion. These sorts of envoys are important in agile teams that need to interact with other groups within an organization [74, 75]. The fact that the envoy was collocated with members from the other team helped in that these things could be communicated quickly and informally. However, the dependency still limited development progress over a number of iterations. To some extent, this development block was out of their hands.

6.3.3 Discussion of Web Components project

During this initial use of XSBD at Meridium, I found many challenges related to communication, collaboration and information sharing issues between team members. A number of these issues related to the fact that the team was distributed. Other issues that were encountered were due to the fact that team members were not used to working with a usability engineer and because the usability engineer was not familiar with working within an agile team. Some of the key findings and corrective guidelines are summarized below:

- Define high-level goals—done in this project through the CDR—that are shared and agreed upon by all team members at the beginning of the project (See Issue 2). This was important to allow the usability engineer to communicate design decisions and their importance to other team members.
- Determine the driving factors of the project and identify their relative priorities. Ensure that the whole team understands what these factors are early on in the project. In this development effort the team was driven more by functional and performance issues than by usability-related issues (See Issue 3). As a result, the software developers had more influence than the usability engineer in dictating project velocity and the design of the overall system. This was not made clear at the start of the project, which contributed to some of the collaboration problems between the usability engineer and the developers at the start of the project.

- For a distributed agile usability team, clearly define communication and information sharing mechanisms to allow non-located team members to remain in sync (See Issue 5). This will require more process structure and documentation generation than is typically expected in agile teams. Developers and usability engineers must maintain regular contact to minimize design drift (emails and regular daily meetings were important here) [51]. Non-synchronous communication mechanisms like email and discussion boards were important to mitigate communication limitations of distributed team members.

At the end of the study period, there were also several open issues that were not adequately addressed and were made part of the focus of the second development effort at Meridium that used the XSBD process

- For teams unfamiliar with the development process—in this case XSBD—some time was set aside before Iteration 0 to introduce agile processes to the team (See Issue 1). Because different team members joined the team at different times, the general introduction to the process did not happen until after work had already begun so the team did not have a common understanding of the process at the start and what each team member's role was on the team. Therefore it is unclear how helpful such introductory measures are.
- The XSBD approach as it was used in this effort was 'agile-centric' in that usability practices were designed to fit within an agile development framework centered on rapid, incremental development. The usability engineer encountered some early problems synchronizing with the developers and had problems consistently delivering designs and running user evaluations because the development team started work early and delivered work at a very high rate (See Issue 4). In the next effort, I planned to explore what synchronization practices and process adjustments were necessary to improve collaboration and validate the 'agile-centric' approach.
- The usability engineer was not able to be integrated within the team as early as expected. This was evident from the very start of the project when developers started implementation work before initial requirements analysis and design work was completed (See Issue 3). As a result, the usability engineer was not able to demonstrate her value to the rest of the team from the beginning, or give the team a better understanding of what she did and why. This may have been because the work that the usability engineer was doing such as goal definition, claims

development, usability evaluations and how they were related through the CDR was not made transparent to the team.

- Factors external to software development and usability affecting project velocity. Meridium have separate QA personnel who are in charge of verifying system functionality. More work needs to be done to integrate QA personnel into the XSBD team so they can work the developers and usability engineer effectively (See Issue 6). Also, the team had a problem with an external dependency to another project team that limited development progress over several iterations (Issue 7). There was a need to determine how best to address these types of dependencies when they are encountered within an XSBD team so it does not slow development velocity.

6.4 Second case study: Touch Screen

The next project at Meridium to use the XSBD process was a standalone touch screen application. This system was complementary to Meridium's overall business strategy in that it was meant to allow customers to obtain higher quality data which customers can then analyze through Meridium's core APM product. This Touch Screen system was a pilot project for one of Meridium's customers and was meant to replace an existing paper-based work order management system at an equipment maintenance facility.

This was the second development team at Meridium to adopt the XSBD process. This project was a smaller standalone effort in that it was not dependent on any existing Meridium software systems. It was also the first time Meridium developed a large touch screen application. The development team for this project was collocated at Meridium's headquarters in Roanoke, VA except for the customer representative.

A total of 252 critical incidents were collected of which 53 were self-reported by the Web Components team members. In addition, a retrospective was held at the end of the release. Finally, I conducted one-on-one interviews with each of the team members at the end of the release. A complete listing of these results is available in Appendix A, Appendix B, and Appendix C

The following section will give a description of the project personnel, the system being developed by the team and then describe findings from the case study. Results from the case study will presented in narrative form with major issues and actions taken highlighted and described in terms of the stages of

action research: What issues are identified (*diagnosing*), what actions were taken to address them (*action planning and action taking*), and how well those actions addressed the issue (*evaluating*). The end of this section will summarize lessons learned and major findings (*specifying learning*).

6.4.1 Project personnel

The project personnel consisted of Meridium employees except for the usability engineer and customer representative. The core team consisted of two developers, one quality assurance person, one usability engineer, one development lead, one product lead and one customer/end user representative. This project was different from the Web Components project in that we had a person from the customer company join the team and act as both the customer representative and end user representative. This person, hereafter simply referred to as the customer, was able to take on both roles for two reasons. First, because he was currently in a management position at his company and was in charge of driving this pilot. Second, because he had experience as a shop worker, which was the type of user that would use the touch screen system. The customer participated remotely from his place of work through regular team meetings and email. The rest of the team members were collocated. The product lead helped to define requirements, and coordinate communications between the customer and the rest of the team members. The usability engineer was recruited from Virginia Tech and had previous experience in interface design and development from previous internships and graduate-level courses in Human-Computer Interaction and Usability Engineering. The team was also assigned a documentation person but he was unable to commit much time to the project due to other project commitments.

The team members on this project were largely unfamiliar with agile processes. I organized several orientation meetings to introduce the XSBD process before the project started and to address any questions and concerns brought up by the team members. In addition, as in the Web Components project, the developers were not accustomed to working with a separate usability engineer. As a result, it took several iterations before the team members got acclimated to using the process and were able to work together effectively.

As in the Web Components project, team members were not fully committed to the Touch Screen project. The developers, QA person and usability engineer averaged approximately 25-40 hours per week on the project. The other team members committed approximately 15-20 hours per week on the project.

6.4.2 Project description

Unlike the first project, the Touch screen project team was largely collocated at Meridium's main office in Roanoke, Virginia. In addition, most of the team, including the project lead, product manager, developers, usability engineer and QA person were assembled at the start of the project. A representative from the client company would soon join the team and work remotely with them in developing the system. The team members on this project were largely unfamiliar with agile methods. In addition, like the Web Components project, they had limited previous experience with working on a team with a usability engineer. To mitigate some of the problems associated with this, we spent some time before development began to introduce the XSBD process to the team and what the responsibilities of each role would be.

Development iterations in this project were two weeks long. Daily meetings were held three times a week since many team members were not full-time on the project. The team primarily kept in contact with the customer on a weekly basis through online-enabled phone conference meetings where they could demo functionality and get feedback. The customer was also accessible through email contact.

After the team came together and was introduced to the XSBD process, they entered the project visioning stage of the development effort (Iteration 0). This would include a visit to the customer site where the touch screen system was to be first installed. It was clear from the beginning that different team members had different motivations and ideas about the project. Having the team define and prioritize goals at the start helped address this issue.

Issue A. Team did not have common understanding of project goals for the system at project inception.

Diagnosis: Like the Web Components project, it was clear that each of the project members had their own motivations and focus points with regards to the Touch Screen project. As shown below, during the first iteration (Iteration 0) where the team was defining the project requirements and preparing to begin development the project lead, developer and usability engineer all had differing and sometimes competing concerns.

The usability engineer prepared questionnaires for some of the customer site personnel to gather data about the end users at the initial site visit. The project lead had some concerns about the questionnaires. The usability engineer noted that:

“<The project lead> and I had a meeting today about the questionnaires I’ve prepared for the client visit. <The project lead> doesn’t think it’s necessary for me to question the clerk and even suggested that we cut back on the manager and mechanic interviews...”

While the usability engineer primarily wanted to gather data about users and the tasks the system was supporting, the project lead and product lead were primarily concerned with meeting and getting to know the customer representative. The project lead and product lead did not want to overly impose on the customer by asking too many in depth questions because part of the purpose of the Touch Screen effort was to build a business relationship with the customer’s company.

Unlike the usability engineer and project lead, the developers were focused on the technical implementation details early on. During iteration 0 the developers were primarily tasked with learning about and setting up the technologies and systems that would be used to develop the touch screen application. A lot of the early discussion between one of the developers and the QA specialist centered on what version of the Meridium connector the team should use. This was a piece of interface software that would allow the touch screen system to store and retrieve data from the core Meridium software.

Action: Like the Web Components project, high-level goals were defined near the end of Iteration 0 as a way to develop a common understanding within the team about what aspects of the system were most important. Unlike the Web Components project, the Touch Screen team focused on defining and prioritizing not just usability design goals, but goals for the project as a whole. This was done to get the entire time to better accept and understand, and be invested in the project purpose. The prioritized goals and associated metrics were defined as shown in Figure 31:

Time Efficiency (*High, not flexible*)
The kiosk data entry must be fast enough to be cost-effective... 2-3 minutes is the maximum input time, and 1-2 minutes is preferred. ...3-5 minutes is acceptable.

User Acceptance of Technology (*Med-high, flexible*)
The user must be able and willing to enter data into our system. We will measure user acceptance along the following lines:
interface makes sense -- can users understand labels, flow of data input, and predict outcomes of actions?

easily learned -- do users get better at quick, low-error rate interactions with the system over time?

low mental frustration -- does the user feel frustrated during and after using the kiosk system?

High Quality Data (*Med-high, flexible*)

The data collected must be of high quality in order to be cost-effective... We will measure data quality according to:

data accuracy -- are all error codes filled out in the digital form that were filled out in the paper form, are comments attached to specific notification items or are they general, is there any useful information on the paper form that isn't in the digital form?

data completeness -- are comments missing from the digital form that were in the paper form, are fewer comments entered now than there were in the original process?

Figure 31. High level goals and metrics for Touch Screen project.

The two highest priority goals for this project ended up being tied to the usability of the system. The end users of this system are highly trained mechanics and technicians so any time they spend not doing actual repair work is seen as suboptimal. Therefore, making the system very efficient to use was the highest priority. In addition, since the success of the system depends heavily on its acceptance by those mechanics and technicians, user acceptance of the technology was the next highest priority goal. The third highest priority goal was to increase the quality of the work data being captured. This goal, which is not directly related to usability, was important in that increasing the quality of this data would allow the client company to better track and analyze that data using additional tools and services such as those provided by Meridium.

Evaluation: Based on post-project interviews, the Touch Screen team members generally did have a common understanding of the high-level project goals. In addition, they agreed that the other members of the team had a good understanding of the project goals. These goals were used to guide many of the major design decisions for the project. The usability metrics were used to verify that the subsequent system met the high-level goals related to the usability of the system (See Issue B). The usability engineer confirmed that the goals:

“helped me keep focus and overall they helped me get the evidence I needed to

open up discussions with teammates about the interface.”

However, having these goals defined and agreed upon at project inception did not address all of the issues as other, undefined project priorities emerged later on. For example, one goal identified later was to minimize information technology costs to minimize customer expense and ensure profitability. One problematic issue that persisted throughout the project was tension between whether to design the system specifically for the customer and designing the system to be generalizable for other potential clients. The usability engineer preferred to design for the end users at the current customer site since they were driving the requirements and she wanted the system to best suit their needs. However, from a business standpoint, the product manager wanted the system to be more flexible so it could be tailored to meet the needs of other future clients. However, it was never clearly defined how flexible the system should be or who the other potential customers of the Touch Screen system would be.

As in the Web Components project, the rest of the team had little knowledge of the role of usability engineer. By having the usability engineer focus on demonstrating her value early and often to the team, other team members were better able to understand what was being done and why. The CDR helped here in focusing usability work, guiding usability evaluations and demonstrating how her work related to high-level project goals. This is described in the issue below:

Issue B. Team has limited understanding of what the usability engineer does and why.

Diagnosis: As in the Web Components project, the members of the Touch Screen project had limited previous experience working with a usability engineer. As a result, they did not understand the purpose of some of the usability tasks and questioned whether they were necessary. This was especially problematic early on in the project when the usability engineer wanted to collect data from the customer and the customer site to better understand the end users of the Touch Screen system and the context in which it would be used. For example, the usability engineer was limited in the amount of data she could collect from the customers during the initial site visit through interviews (as mentioned in Issue C). In addition, during the first iteration the QA person and product manager questioned why the usability engineer wanted to show early mockups to the customer. The QA person did not want to show an ‘unpolished’ UI to the customers because she believed it would make the company look like it was not competent. They did not understand the importance of using such mockups in an agile usability method like XSBD to get early and continuous feedback from customers

and end users. They were familiar with more traditional development methods where customer feedback is more infrequent and tightly controlled.

Action: There were meetings at the start of the Touch Screen project to introduce the XSBD process and the usability engineer. Unlike the Web Components project, the introductory meetings were run at the start of the project with the entire team present. Although they were useful in giving a high-level overview of the approach, they did not give the other team members a good idea of what a usability engineer actually does and how they can add value to the team—something we also noted in the Web Components project (See Section 6.3.2: Issue 1). In fact, such meetings can be seen as an annoyance to other team members. One member noted that she felt like she was being lectured to. As a result, we focused on presenting the tangible results of usability work as early as possible to other team members. These tangible results are those parts of the usability work that are directly related to the design of the system including the development mockups and usability testing results. The usability engineer always focused on presenting these results in the form of the CDR—naming showing how design features were related to high-level design goals and how they were validated through specific usability tests (See Figure 32). This way, the usability engineer could explain why features were designed a certain way and present evidence to show their efficacy in meeting those goals. Other usability work related to how and why that tangible work was done (e.g. claims development, evaluation planning, etc) was not generally shared with the team unless they inquired about it.

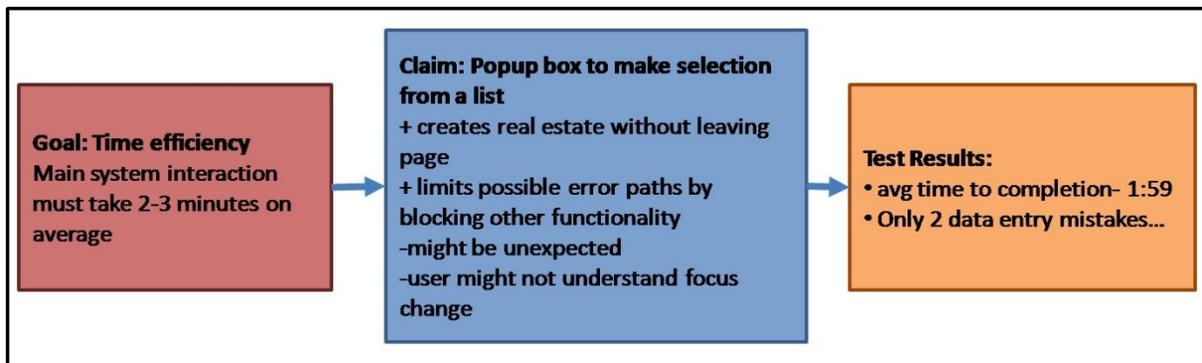


Figure 32. Conceptual image of how the usability engineer communicated results. Each design claim (middle) is linked to a high-level goal (left) and is validated through usability tests (right).

Evaluation: By showing the results of usability engineering work, the team was able to better understand why that work was being done and how it was valuable to the team as a whole. In fact, by having the usability engineer work within an agile development framework where parts of the system

and design were developed and tested in each iteration, she was able to more immediately see the impact of her work in the process and show other team members the impact that usability was having on the overall design. For example, after showing mockups for the first iteration to the team, the usability engineer noted that:

“I’ve been getting positive feedback about my UI mockups. In the meeting, <the project lead> and others said good things about the design. In my office, <the developer> stopped by and said that the design was excellent...”

In addition, comments during the retrospective held at the end of the release showed that the team had a better understanding of usability-related activities and their value to the overall project:

9/17/08 Retrospective

What went well:

- Incremental mockups allowed us to show customer that his feedback got reincorporated into product. (Customer feels ownership of product)
...
- UI Design decisions could be guided by usability tests.
- UI Design decisions provided information that customer didn’t necessarily know. (I.e. sometimes the customer didn’t know if one design was better than another).

This is in contrast to other proposed integration methods such as that initially proposed by Cooper where usability requirements and design work would be done first before traditional agile development could proceed [90]. In such an approach, the impact of usability work would not be seen by the team until much later in the process. In addition, much of that initial usability work might become stale as new and changing requirements are uncovered.

The specifics of how the usability engineer was doing her work did not appear to be of primary importance to the other team members. For example, usability artifacts such as the claims the usability engineer used to document important features of the system and to guide usability tests were uploaded to the team portal on a regular basis but were largely ignored by the team. The primary way that usability work was being communicated was through the tangible artifacts mentioned above. Similarly, the usability engineer only viewed the developers’ work in terms of the functioning code that was delivered. She did not actually view the code itself or discuss in detail how the developer engineered it. In this sense, the team only communicated the information necessary for each member to do their own work.

Meridium includes a number of different teams—most of which were not using the XSBD approach—that work on different applications targeted to different customer groups. As such, the Touch Screen team found that their work was sometimes dependent on factors external to the team. The impact of this issue and how it was addressed is described below:

Issue C. External requirements not communicated clearly to the entire team resulting in disagreements and wasted work.

Diagnosis: Early on in the project, requirements driven by company-wide initiatives external to the team itself were not clearly communicated to the entire Touch Screen project team and resulted in certain miscommunications and disagreements. For example, during iteration 0, the team was planning a site visit to the customer. The usability engineer was planning to use this opportunity to do some initial requirements gathering through observation of end users and interviews. However, upper management at Meridium wanted to use the site visit as a way to build the relationship with the customer. Upper management did not want the usability engineer to do any interviews or observations as initially planned, as they did not want to overwhelm the customer. This issue did not come to the attention of the usability engineer until the day before the site visit. The product manager had communicated his concerns to upper management directly without discussing concerns with the rest of the Touch Screen team. Similar problems came up with regard to what underlying technology the developers would use to implement the system and how the system itself would communicate with the underlying core database.

Action: As in the Web Components project, the Touch Screen team established clear lines of communication between external parties that had influence on the team. For example, the product manager and project lead would be the contact points between upper management and the rest of the team. Also, one of the developers was established as the contact point between the Touch Screen team and with two separate external entities: the core database team at Meridium and the database team at the client company. This was necessary as the touch screen system would have to be compatible with the core Meridium product and interface with the customer's database when the system was delivered. Each team member who was a contact point with an external party would keep the rest of the team up to date and communicate issues and concerns between the two groups as necessary. These external communication issues were resolved through a series of meetings between team members—typically the regular daily meeting or weekly feature team meeting.

Evaluation: Establishing the contact points and more clearly communicating how these external issues affected the team addressed many of the problems that arose. For example, at the start of the first iteration, the project lead revealed that upper management determined that the system would be implemented as a web application and explained that it was strategically important for Meridium because it would help minimize installation and maintenance costs. The developer noted that:

“<Product management> communicates their new requirements more clearly and even through the architecture they propose has potential issues, it fits the company’s general vision.”

However, this did not entirely mitigate the problem of external influences causing confusion within the Touch Screen team. For example, there were disagreements throughout the project regarding whether the project should be designed to closely match the needs of the client company or should be generalized to potentially fit the needs of a broader range of customers. The usability engineer wanted to develop the system to closely match the needs of the current customer rather than design for hypothetical future customers. However the project manager thought that the system should be designed so that it could be sold to a broader range of customers. Despite having a contact with upper management, the team did not get any clear direction in terms of which approach to take. This problem may be related to the fact that the product management team at Meridium was not able to do comprehensive background/market research before the project started.

One open question at the start of the Touch Screen project was how to best communicate usability work with other team members. Unlike the Web Components project, the team was collocated and so would likely not need to depend as much on shared documentation. In addition, special focus was given to how to optimize the type of information shared between the usability engineer and the rest of the team so they would not be inundated with information they did not need to know and so the usability engineer would not have to spend excess time developing documentation that wasn’t needed. This issue is detailed below:

Issue D. How to best communicate usability work (CDR, design documents, evaluation results, etc.) so the team can operate effectively.

Diagnosis: One open question at the start of the project was how the usability engineer should share and communicate her work to the rest of the agile usability team so they can work most effectively. Previous work has shown how the CDR can be used as a bridge between usability and development to

help in sharing design work and design rationale [64, 67]. This preliminary work was done by student developers working as an XSBD team in an academic setting. One problem this work is that the CDR itself may require too much effort to develop and maintain which would hinder its adoption among agile practitioners [20]. In the earlier Web Components project, the usability engineer did not explicitly share aspects of the CDR and relied more on verbal communication. This may have contributed to the problem of the usability engineer being ignored at times in the Web Components project—as she did not share the work that she was doing to other team members.

Quick Usability: Iteration 0
your quick guide to the status of usability

High-level goals
the prioritized list of usability goals

- time efficiency (*High, not flexible*)
 faster than current process to the point of cost-effectiveness
- user acceptance of technology (*Med-high, flexible*)
 interface makes sense
 easily learned
 quick to complete
 low mental frustration
- high quality data (*Med-high, flexible*)
 accurate data
 complete data
- smooth workflow (*Medium, flexible*)
 minimal modification to current efficient practices
 minimal addition of new practices to fill in gaps

Priority Features
the prioritized list of features to develop to achieve our goals

- Vertical Scrolling to Navigate Content
there should be a vertical scrolling mechanism to navigate the form. no horizontal motion, no separate screens (i.e. no forward and back buttons)
 - + *there are only two places the content can be: up or down*
 - + *is commonly used in most prevalent applications*
 - + *there is no issue with returning to previous screens*
 - + *no anxiety about losing data or not being able to change entered data after progressing to another screen*

- + *follows the form metaphor; manipulate the digital form in the same place as the physical*
- *might obstruct view of toggle action*
- *user might not know that this area is 'hot'*

Usability Results
the results of usability testing of our developed features

 this will be a summary of the results of usability testing at the end of each iteration

Other Resources
for all your other usability needs...

[What We Know About the User](#)

[Usability Scenarios](#)

Figure 33. Document summarizing usability work. Contains high-level goals, design claims, usability results and links to other resources.

Action: In the Touch Screen project, the usability engineer explicitly shared more of the work she was doing but only explicitly shared aspects of her work which could be easily understood and used by the rest of the team. In terms of the central design record, this meant focusing on the *high-level design goals* for the system, the *design mockups and specific design features* of the proposed design, and *usability testing results* that showed how the design met the high-level goals. For example, at the end of Iteration 0, the usability engineer presented a ‘Quick Usability’ document which summarized initial requirements engineering work she had completed after the initial customer site visit (See Figure 33). This document was maintained throughout the project and contained high-level project goals, critical design features and claims and summaries of user testing results. This document was provided by the usability engineer to give the rest of a team a single place to get a high-level overview of what she was working on.

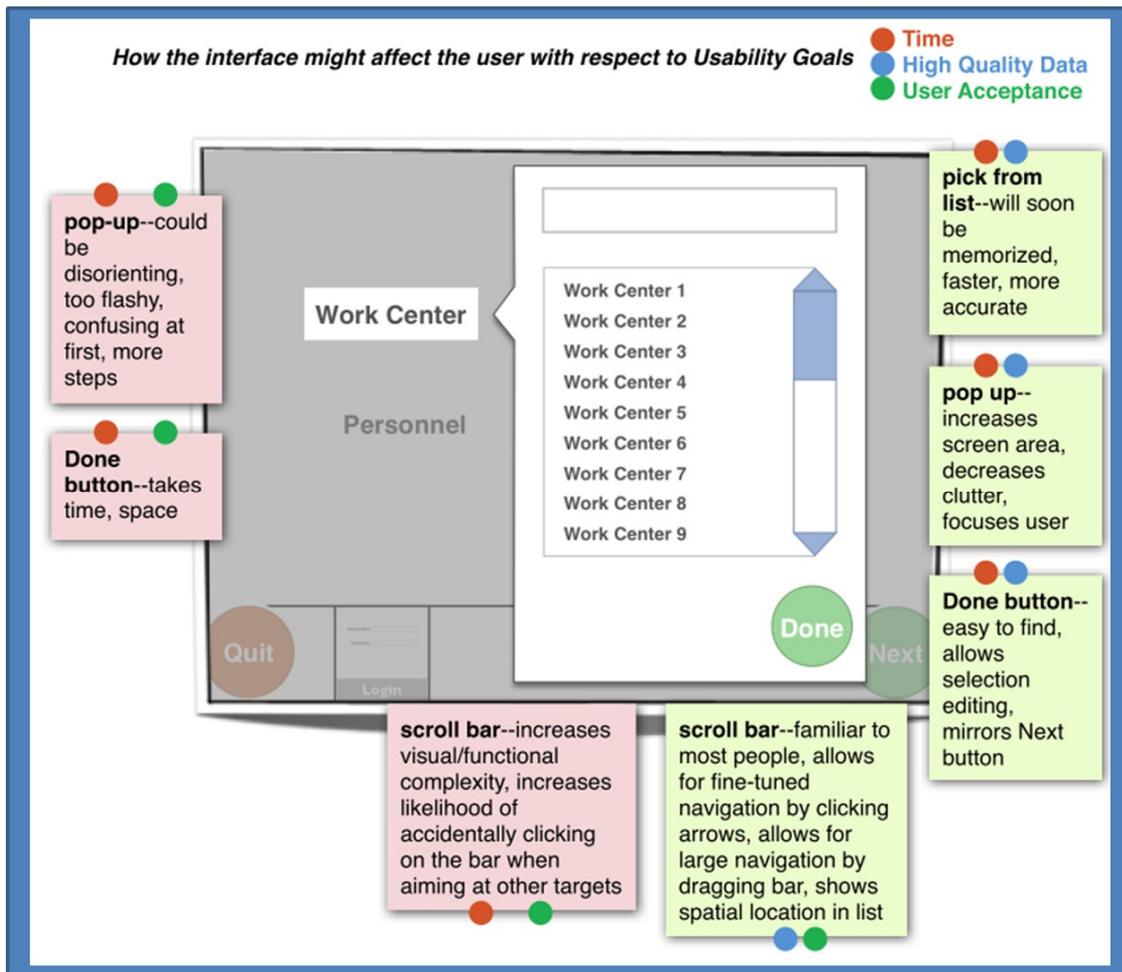


Figure 34. Design mockup with associated claims. Downsides are shown in pink while upsides are shown in green. Claim relationships to high-level goals are shown using color coded dots.

In addition, the usability engineer tightly associated design claims to the prototypes by showing claims superimposed on the prototypes she designed (See Figure 34). In addition, a color coding scheme was used to show how each claim related to one or more high-level goals. This representation was primarily used by the usability engineer to track the relationship between important design features and goals and was later used for usability evaluation planning. However, it was also regularly posted to the shared document portal so that the rest of the team could review them.

Evaluation: In terms of sharing designs and rationale for design decisions, the usability engineer ended up relying primarily on verbal communication and the actual design mockups she put together. For example, QA relied extensively on the design mockups to do test planning. Team members in general noted that they did not often look at the other design documents the usability engineer maintained such as the claims or even the quick usability document. Rather, they relied more on her sharing her designs and verbally communicating things like user testing results. At the end of the first release of the Touch Screen project, the project lead noted that:

“To do her job, she needed to connect the usability to her design decisions and that’s good. But I don’t think that sharing all that information with everyone was necessary.”

In fact, the most effective way that she demonstrated her value to the team was through sharing the design mockups and usability testing results with the rest of the team—demonstrating how they related back to the high-level project goals. This is the critical part of the CDR we determined needed to be shared with the rest of an XSBD team. The specifics of the central design record—such as the design claims she was using to plan and run those tests—were important in that they helped the usability engineer work with the rest of the agile team. However, they were of less relevance to the rest of the team and did not need to be shared on a regular basis.

In XSBD the developers and usability engineering groups work in parallel which requires careful coordination for each group to work effectively without slowing development velocity. In practice, the parallel work process and handoffs did not occur exactly as planned and required some flexibility in terms of how the two groups planned and coordinated their work. This is described in detail below:

Issue E. The usability engineer and developers are not able to work in parallel effectively.

Diagnosis. A key characteristic of the XSBD process is for the usability portion of the team and the development portion of the team to work synchronously. Figure 35 shows an abstraction of how the

Touch Screen project would have progressed if it had gone according to plan. During iteration 0, the team would conduct an abbreviated requirements analysis process. During iteration 1, the software developers would complete non-UI related work such as exploration/research of implementation technologies and environment setup. The usability engineer, in the meantime would work on designing the first portion of the design (D1). In iteration 2, the usability engineer would handoff the initial design to the developers to start implementing while working on designing the next portion of the interface (D2). In iteration 3, the developers would handoff the implementation of the first design (ID1) to the usability engineer, who would evaluate it [$E(ID1) = D1'$]. The developers would then begin work on implementing the next design (ID2). As the Touch Screen project is a relatively small effort, the first release—which consisted of the software that would run on the Touch Screen—would take only five iterations to develop. In iteration 6, the entire design of the first release would be evaluated by the usability engineer [$E(R1) = R1'$].

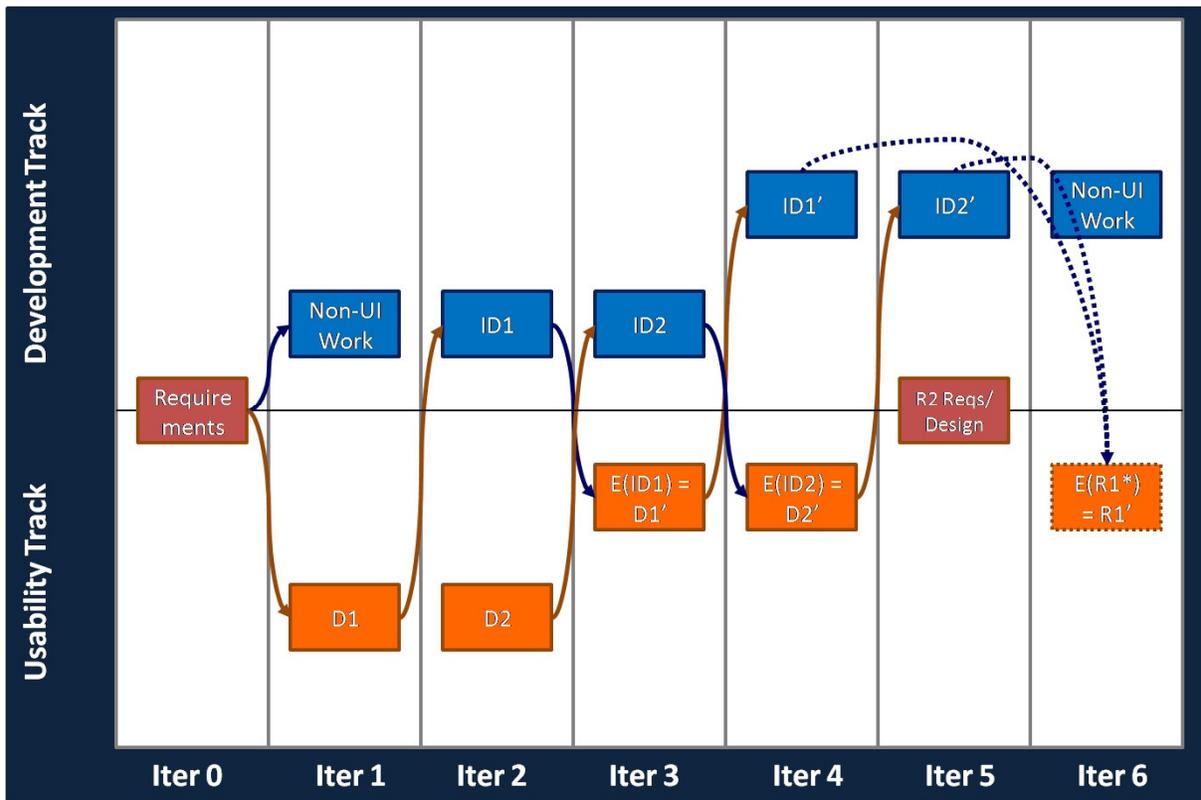


Figure 35. Planned work synchronization based on XSBD process.

As the Touch Screen project progressed, the ideal synchronous handoffs between the usability engineer and the developers did not occur as originally envisioned. This was due to a number of different reasons:

- The Touch Screen team started work later than planned because the team could not be put into place due to time and resource constraints related to other projects.
- Team members were not experienced with agile methods. In the early iterations, the project lead had difficulty planning iterations because the team was not sure of the amount of work that could be completed within an iteration and also whether all development, QA testing and usability work for an iteration could even be done within a single iteration.
- The team was not used to working with a usability engineer (See Issue B), so it took some time for the usability engineer to be able to work within the team effectively. In particular, it was initially unclear how the usability engineer would effectively plan and run usability evaluations so that results could be handed off the developers by the start of the following iteration.

Action. It was clear early on that there needed to be some flexibility in how the handoffs between the usability and development sides of the team would occur. The actual handoffs as they occurred are shown in Figure 36. Although it may at first glance appear to be far different from the ideal XSBD process, the basic structure is preserved whereby the usability engineer designs the user interface while the developers implement the designs.

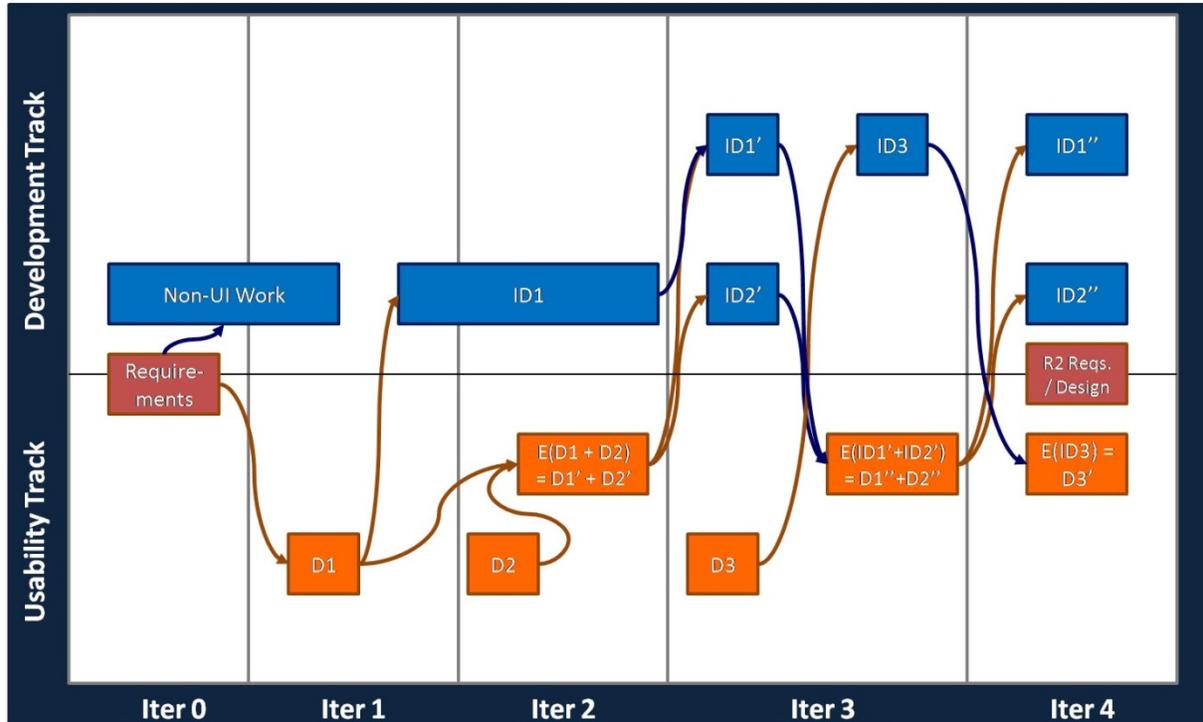


Figure 36. Actual work synchronization. Note that horizontal widths of iterations vary only to better show contents of each. Each iteration covered the same amount of time.

In iteration 0, the developers worked on Non-UI related work starting early on—focusing especially on learning and evaluating the underlying software technology that would be used to implement the Touch Screen system. During iteration 1, the usability engineer was able to complete the design for the first portion of the interface (D1) early, which allowed the developers to begin implementing it immediately. It was originally planned to have the developers finish implementing the first design (ID1) by the start of iteration 2. In iteration 2, the usability engineer would design the second portion of the interface (D2), and run an evaluation of the delivered work (ID1). However, the developers were not able to deliver the implemented design on time due to difficulties related to the software technology being used. To get around this difficulty, the usability engineer ran a usability evaluation using a 'click-through' prototype she put together based on her designs (D1, D2). This 'click-through' prototype was a series of images from her design tied together with hyperlinks and allowed users to run through the primary scenario of use. This way, she was able to hand off redesign recommendations to the developers at the start of third iteration for both the first and second designs (D1', D2'). In iteration 3, the developers worked on implementing the changes proposed by the usability engineer and were able to deliver them (ID1', ID2') near the start of that iteration. They were then able to work on implementing the third design, D3, which the usability engineer delivered midway through the iteration. The usability engineer was now able to run a usability evaluation of the actual functioning system [E(ID1' + ID2')] which resulted in additional recommended changes [D1'', D2'']. In the fourth iteration, the developers focused on implementing the changes based on the last evaluation (ID1'', ID2''). The usability engineer evaluated the next set of functionality [E(ID3)]. In addition, she worked with the product manager to come up with requirements for the next release (not covered by this study).

As described above, the team did not work exactly within the handoff process described in XSBD. The team worked on a somewhat compressed schedule because they began work later than expected. The usability engineer was able to work within this tight schedule through a combination of quick design mockups and focused, lightweight usability evaluations (described in detail in Issue G). This was all managed explicitly including usability tasks into the iteration planning process. Thus within each iteration, both the usability engineer and the developers would select and estimate tasks. This can be contrasted with the Web Components project where the team only accounted for developer tasks when tracking iterations.

Evaluation. By having a unified planning process the team had a better idea of what both the developers and usability engineer were working on throughout the project. This made it clearer what

things would be delivered within each iteration and what handoffs could or could not be made. The developers and usability engineer were able to then make adjustments as issues were uncovered during the daily standups or weekly team meetings (e.g. when the usability engineer had to develop a click-through prototype to run an evaluation).

This issue and how it was resolved shows how the XSBD process can address different difficulties related to how handoffs between the usability engineer and developers occur. In addition, such flexibility is important, as similar difficulties are common in software projects [12, 45, 108].

Even when the usability engineer and developers were able to synchronize their work processes at the start and end of each iteration, there were still synchronization problems within each iteration. The team had to develop strategies to work more closely within each iteration to prevent drifting between designs and implementations.

Issue F. Usability engineer and developers have problems staying synchronized within iterations.

Diagnosis: It became clear during the project that additional steps needed to be taken other than those described in Issue E for the usability engineer and software developers to work together effectively. Even with the handoffs described there, problems such as inconsistencies between the design and implementation and general disagreements about the design between team members could hinder progress.

One problem in the Touch Screen project was related to the fact that the team was not accustomed to working with a usability engineer. In typical projects at Meridium, the user interface would be designed by a combination of developers and project managers with input from all other team members. One problem with this ‘design by committee’ approach is that the UI design can become unfocused and not meet the needs of the end user [38]. Having a dedicated usability engineer on the team whose responsibility was designing and maintaining the UI was meant to avoid this issue. However, the usability engineer felt that other team members were trying to influence the UI design and they were not trusting her to do the work herself. The project manager also noticed the problem noting that:

“Some team members were trying to make requirements/functionality/usability decisions without data to back those decisions up... can lead to developing the wrong functionality.”

A second consistent issue throughout the project was that implementation would not match the UI design. Sometimes this was due to technical issues. For example, the developer added an animated ‘wait’ animation to one of the screens because of the extended time it took to poll the backend database before being able to update the screen. At other times, the UI design would change within an iteration based on feedback from the customer representative or because of holes in the design that were uncovered by other team members. Developers would get frustrated as they would then have to rework code they had already started on when given these updated designs.

Finally, at times, interpersonal tensions between the usability engineer and developers prevented them from working together effectively. These tensions came about partially as a result of the two points noted in the above paragraphs but also because of personality conflicts between them. One developer in particular would often disagree with the usability engineer about how the system should be designed and would implement the system to his own specifications instead of those provided by the usability engineer. This would cause problems because the developers would do this work without consulting with the usability engineer and the issue would only come to light during a demo of the system at a team meeting. One QA person noted after an especially tense meeting:

“Today's standup meeting was difficult to sit through because the first "final-looking" version of the touch screen project was unveiled...and it wasn't what usability had intended for it to be. There were many conflicting opinions concerning it and at times there was a lot of contention.”

Action: The team took several steps to address the synchronizations problems noted above.

First, as the usability engineer established the value of her work (iteratively designing the interface and validating that they meet goals through usability testing—see Issue B), the team established a methodology for dealing with questions or disagreements about the design. If another team member thought a change should be made to the design but no compelling rationale was given for why the change should be made, the team would go with the original UI design from the usability engineer first. The alternate design would be recorded—typically in the form of a claim—and the usability engineer would evaluate that portion of the design in her next usability evaluation. If a problem with the original design was uncovered, then the usability engineer would use the alternate design proposed by the other team member assuming it addresses the problem appropriately.

Second, the usability engineer and developers began collaborating more closely with one another within each iteration. This included more day-to-day interactions to address questions or issues that other team members had with the UI design or with the implementation of that design. One tester noted that:

“After our normal stand-up meeting, a few of us got together to discuss questions and concerns about the User Interface [design]. We were able to be open about suggestions and it facilitated communication especially since it helped us focus on perfecting one aspect of the project. We remained in contact via email after the meeting to give feedback on changes and iron out the details”

Such meetings became more commonplace, and especially helped to address some of the disagreements between the usability engineer and the developers. Similarly, the developer made an effort to demo the system to the usability engineer as it was being implemented so the usability engineer could review it and voice concerns about the UI if they differed from her original design. In the later iterations, the usability engineer relied less on trying to exactly define the UI design and all possible interactions with it. Using this ‘mid-fidelity’ prototyping approach, the usability engineer relied on these regular interactions with developers within each iteration to flesh out areas of the design that had not been completely specified. This approach seemed to work better given the limited time team members could devote to this project and the various issues that could arise with respect to how closely the implementation matched the design.

In addition, for any subsequent instances where the implementation did not match the design at the end of an iteration, the usability engineer would document the difference in the form of a claim and evaluate it in her next usability evaluation. If the evaluation showed that the implementation had problems, then the developers would implement the original design specified by the usability engineer. This helped to diffuse conflicts between the usability engineer and developers and further demonstrated the value of user testing to the rest of the team. One example of this situation was in the start page of the Touch Screen application (See Figure 37). The usability engineer originally specified that this page display a simple green ‘start’ button to provide users with an easy way to begin using the system. One of the developers instead implemented a rotating animation that showed the green ‘start’ button on one side and an image of the red Meridium logo on the other. This was done to satisfy the branding requirement of the product. The usability engineer thought that this animation might not make it obvious that the animated icon is clickable and might be annoying to see constantly. She

documented the issue in the form of a claim and evaluated it in a user test. The tests found that the animated icon did not impede users so the design was kept as it was implemented.

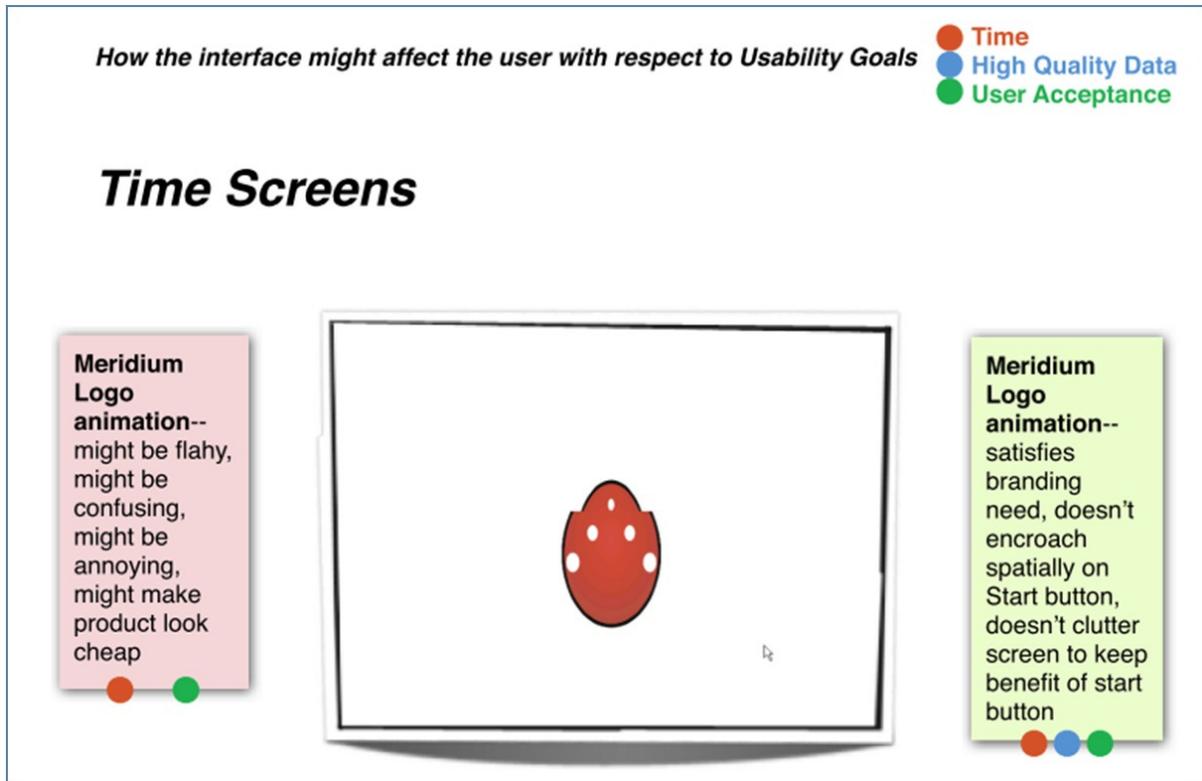


Figure 37. Claim describing start page logo. Downsides are shown on the left and upsides are on the right.

Finally, to reduce developer frustrations with constantly changing UI designs, the UI design for each iteration was frozen at the start of the iteration. Any proposed changes or improvements that were not critical were moved to the backlog for inclusion in the next iteration.

Evaluation: The problems described in this issue highlight some of the challenges of having the usability engineer work in parallel to the developers. Although there are benefits in terms of the overall velocity of the team, significant effort is required for both to remain synchronized throughout the development effort. The mitigating actions showed how close collaboration between the usability engineer and developers within each iteration and guided by the CDR—and claims-based usability evaluations in particular—can address some of those problems. This particular project also highlights the central importance of the people involved in each project and how they work together. Focusing on the people involved and their collaborations is a central tenet of agility and is vital to the success of XSBD-based teams as well [13]. By the end of the project, the usability engineer and developers were working together more effectively. The developer noted in the post-project interview that:

“It’s great that the developer doesn’t have to worry about how the UI is going to look like. When you are developing from scratch and you have time constraints, and you have a design, then you can do it efficiently. ”

Similarly, the general consensus from the retrospective at the end of the release was:

9/17/08 Retrospective
What went well:
...
- Having <a> usability engineer had a huge positive impact. Allowed developer to focus on implementation/design issues instead of trying to handle UI issues.

This demonstrates his acceptance of the role of the usability engineer within the team and the value she provided.

Although the CDR provided some guidance to the usability engineer in terms of what parts of the interface needed to be evaluated and why, she still needed to decide what types of evaluations to run given the time and resource constraints of working within an agile framework. This issue is described below:

Issue G. Usability engineer had difficulty running usability evaluations within agile framework.

Diagnosis: As in the Web Components project, the usability engineer had some difficulties planning and running usability evaluations throughout the project. This was because of several different reasons. First, since the other team members did not previously have experience working with a usability engineer, they did not have a good understanding of the amount of time required to do things like plan an evaluation. For example, early on the project manager expected the usability engineer to be able to plan and run an evaluation the next day. The usability engineer explained that she would be unable to do this because she needed more time to prepare and recruit participants. Second, the usability engineer was unable to recruit end user participants at the customer site for any of her evaluations before the end of the study period because of bureaucratic issues. Third, the usability engineer had to deal with certain technical and process-related issues. One example of this was in the second iteration when the developers were unable to deliver the implementation of the first design on time—which meant the usability engineer was unable to run an evaluation of it. Another example was in the third iteration, when the physical touch screen itself had still not been delivered to Meridium,

making it impossible for the usability engineer to evaluate the implemented design as intended. Finally, the accelerated timeline and fast-paced development cycle of the Touch Screen team meant the usability engineer had only limited time within each iteration to recruit participants, run the evaluation and analyze the results.

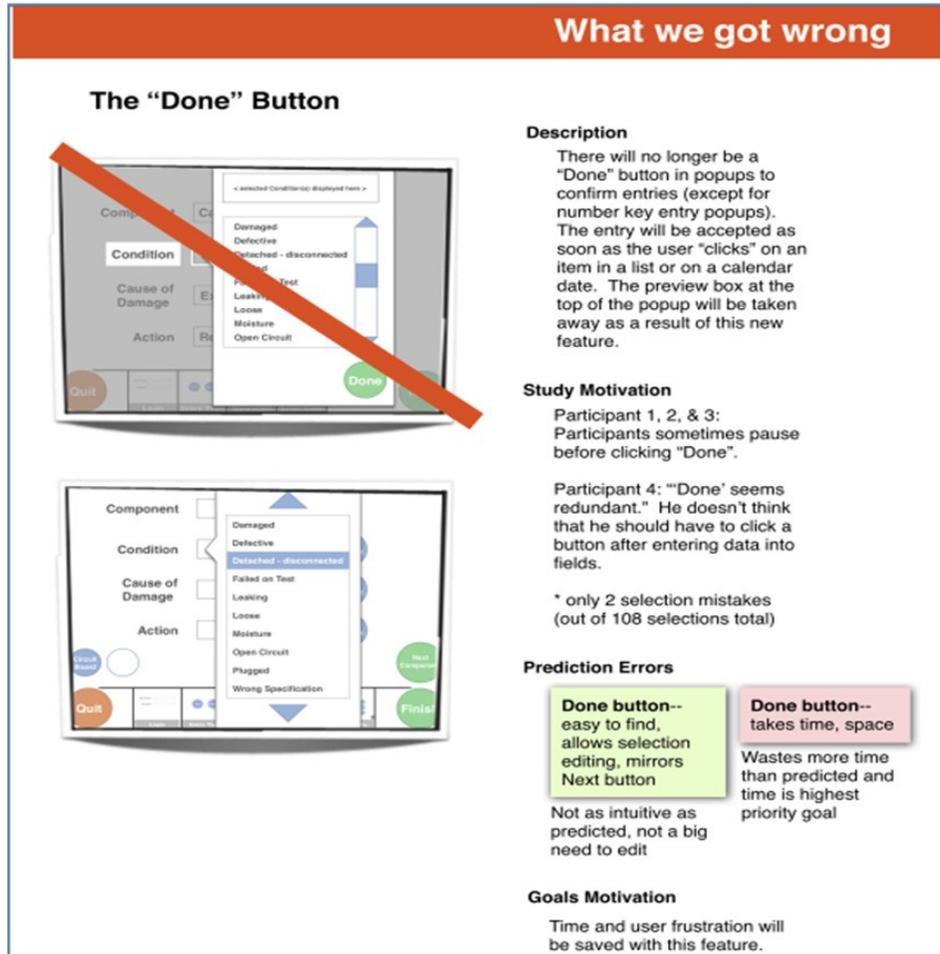


Figure 38. Usability evaluation results showing the design recommendation (description), evidence from the study (study motivation), claim tradeoffs (prediction errors) and related high-level goals (goals motivation).

Action: The usability engineer took several different actions to mitigate the problems outlined above and run evaluations so that she was able to provide timely, actionable feedback to the rest of the team.

First, she relied on claims-centric evaluations—a key part of the CDR—throughout the project. That is, as she designed each piece of the interface, she would document claims that related to features that most closely aligned with the high-level goals of the project (See Figure 32). These claims could also arise from team discussions or customer concerns. In the following iteration, she would plan her

evaluation so that she would be able to verify the claims she made about the design (e.g. are the upsides of a claim true?). Based on the results, she would then verify that the design is meeting high-level goals or identify areas that needed to be redesigned so those goals could be met. Figure 38 shows evaluation results related to one of the claims from the second iteration. This particular result shows that the 'Done' button on selection popups should be removed. The original claim upside (shown under Prediction Errors in Figure 38) stated that the 'Done' button is "easy to find, allows selection editing, mirrors Next button". However, her study showed that the button was not as intuitive as originally predicted. In addition, the downside that the button "Takes time, space" turned out to be true. As this related to the highest priority goal of time efficiency, this change was given high priority and was implemented in the next iteration.

Second, the usability engineer relied on lightweight or discount evaluation techniques [93]. These are methods that sacrifice some quality of the usability evaluation results for increased speed and efficiency in obtaining them. The usability engineer reduced and focused the amount of the interface being evaluated, as described in the previous paragraph. In addition, each evaluation involved a relatively small number of participants, typically 3-5 people. Studies have shown that even this small number of participants can yield good results [93]. In addition, the participants themselves were recruited from within Meridium. The usability engineer recruited participants based on how closely they matched the profiles of the end users she had created in Iteration 0 according to parameters such as computer literacy. She additionally verified her designs based on general touch screen design heuristics obtained through online research and expert feedback from someone at Meridium who had previous experience designing touch screen interfaces.

Third, the usability engineer had to be able to deal with situations where she would not be able to run evaluations with a functioning system. For example, when the developers were unable to deliver the implementation for the first design on time, she developed a 'click-through' prototype to run the evaluation (described in Issue E). In another instance, the touch screen hardware was not delivered in time so the usability engineer was unable to use it in the evaluation she planned during the third iteration. Instead, she relied on a 'Wizard of Oz' technique [111] where she ran the software on a flat-screen monitor and had users touch it as if it were a touch screen. The actual system was being controlled by another team member who was using the computer mouse to simulate the participant's actions.

Evaluation: When working within an agile team, the usability engineer was unable to run tightly controlled, large scale usability evaluations as is common in many usability processes due to the fast pace of the team [111]. Rather, the usability engineer had to run small-scale, targeted evaluations guided by the CDR in combination with lightweight evaluation techniques. This allowed the usability engineer to obtain good (but not optimal) usability results that she could analyze quickly and turn into actionable design guidance tied to the high-level goals. This allowed the developers to place these feature redesigns into the product backlog after providing development estimates. In addition, in keeping with the agile mindset fostered by using these techniques, the usability engineer was able to deal with a number of problems that initially blocked her from running evaluations. Finally, she found that even when working within the incremental development cycle and running a larger number of smaller scale studies she was able to:

“...get a decently consistent interface that fits the customer better.”

That is, the usability engineer believed that she would have gotten a more consistent and cohesive design if she had been able to do a more complete requirements analysis and design phase at the beginning of the project, but in the end thought that the resulting design fit the customer’s needs.

However, one cannot discount the value and importance of running controlled, larger scale usability evaluations. These evaluations, which might involve large numbers of end users running through task flows that cover several iterations worth of work, can help to verify redesigns made based on past evaluations and provide more complete and comprehensive feedback to the team [111]. The original plan was to run larger scale evaluations at the end of each release cycle. Unfortunately, due to some technical issues and difficulties related to recruiting participants, the usability engineer was unable to run such an evaluation during the study period. Therefore, the impact of running such studies on a team using XSBD is unresolved.

One issue that came up was that even though there was someone on the team from the customer company who could act as both the customer representative and the end-user representative, he did not always know what the best decisions were in terms of the UI and specific features to implement. The usability engineer was helpful in translating user needs into a design and demonstrated her value to the other team members. This is described in detail below:

Issue H. The customer is often unsure of the optimal decision in terms of features and design.

Diagnosis: Agile methods such as XP presuppose an onsite customer whose responsibilities included defining and prioritizing features and doing acceptance testing of features once they are implemented. Agile practitioners including Jeff Patton have noted that the shortcoming of this approach is that at times customers do not know what the best feature should be or whether the design as it has been implemented is sufficient [56, 102]. This problem became apparent in the Touch Screen project early on. Although the customer/end user representative was knowledgeable about the processes the touch screen system was meant to support and had previous experience using it, he did not have much knowledge of touch screen computer systems, their capabilities, and how they could best be implemented within the work practice.

Action: The usability engineer worked closely with the customer to ensure that the design of the UI would work best for the end users. This was done largely through communication of targeted usability evaluation results to the customer. If the customer had a concern about the way a feature was implemented during one of the iteration review meetings and if the concern was related to the usability of the interface, then the usability engineer would note the concern and review it against her rationale for the design. Depending on the scope of the proposed change and the strength of her existing rationale, she would capture that concern in a claim and evaluate it.

For example, one later design decision that had to be made was whether the system should use a physical keyboard or an onscreen keyboard for data entry. Some team members argued that the physical keyboard would be easier to learn and support faster data entry. The usability engineer argued that the onscreen keyboard would reduce the number of data entry errors. One of the developers also argued that the onscreen keyboard would take less time to implement. In fact, the customer expressed a slight preference for a physical keyboard. The usability engineer ran an A-B evaluation comparing the use of the physical keyboard and onscreen keyboard. She found that users were able to complete tasks more quickly with the physical keyboard but had higher error rates because users switched their focus between the touch screen and the keyboard when inputting data. Although data entry with the onscreen touch screen was slower, error rates were lower and the total task time still met the high-level efficiency goal. When presented with this data, the customer agreed that the onscreen keyboard was the better option.

Evaluation: The customer, and the team in general, came to rely on the usability engineer to answer questions related to the interface itself. Specifically, they came to trust usability design decisions that had been validated through usability evaluations. The product manager noted that:

“Anytime you have a team saying ‘I think...’ or ‘Why don't we...’ they are making decisions without data to back up that decision... You must have data to base your decisions on. Usability decisions should be based on user studies “

This again demonstrates the acceptance of the usability engineer and shows how design decisions guided by high-level goals and validated through targeted lightweight evaluations can be of value within an agile team. It shows how the usability engineer can help the customer/representative end user instantiate the system so that it best meets end user needs.

One shortcoming that was not optimally addressed with the Touch Screen team was how to integrate QA and documentation personnel into the XSBD team. This was partly due to the fact that the team did not have adequate documentation and QA resources that could be dedicated to the team. It was also due to the fact that these areas are not the focus of this work. This problem is described in the two issues below:

Issue I. Team did not effectively integrate QA processes into XSBD.

Diagnosis: The project team lead wanted to more closely integrate QA into incremental development process by having a complete implement-test-fix cycle within a single 2-week iteration. That is, he wanted developers to complete their feature tasks and then hand off the implementation to QA who would do a first-pass test of the completed code. Any bugs that were found would then be sent back to the developers to fix. In addition, he wanted usability testing of that implemented code to be done within that same iteration. This concerned the usability engineer as she noted that:

“<The project lead> initially said that all feature rework must take place within one two-week iteration and that a feature will be considered complete by the end of the iteration. This concerned me greatly because this leaves no time for usability evaluation and resulting changes.”

It proved difficult to have the QA tester and the developers work synchronously within a single 2-week iteration. In fact, during the first iteration, the developers were unable to deliver work in time for the QA tester to begin verifying the implementation. Another problem related to synchronizing efforts between the usability engineer, developers and the QA tester. For example, when the QA tester began verifying the implemented code, he would document bugs and then assign them back to the

developer. This was problematic as some of the changes had a non-trivial impact on the user interface and the usability engineer was unaware of them.

Action: Unlike the Web Components team, where the offsite and onsite developers were initially using different work management processes, this team used a single unified work tracking system. As this team was collocated, their coordination difficulties did not stem from the fact that the team was distributed. Rather, the team was unable to adopt the implement-test-fix cycle initially proposed by the team leader because of the relative inexperience of the team with respect to agile methodologies and general organizational resistance to working within such an accelerated development cycle. The additional complication of having a usability engineer on the team (with her own work practices, deliveries, etc.) made such coordination even more difficult.

The team took the following actions to address the issue of how to integrate QA into the XSBD process:

- 1) From the second iteration onwards, the team decided that the development cycle would be relaxed. Now, during the first week of an iteration, the developers would work on new features that would be delivered to QA by the start of the second week. QA would then start testing that first week's work in the second week of the iteration. In the meantime, the developers would work on other feature development or rework from QA tests or usability evaluations. In this way, the developers and QA began working in parallel—similar to how development worked with the usability engineer—with the developers working one week ahead of QA.
- 2) QA began working more closely with the usability engineer through regular weekly meetings and additional impromptu meetings as necessary. These meetings allowed the usability engineer to show and explain the mockups and scenarios to the QA tester at the start of each iteration. QA then used this information to develop the test plans that would be used when the working code was delivered. In addition, these meetings were important in that the QA tester would often uncover holes in the design (e.g. error conditions, error messages, and 'corner cases'—non primary scenarios) that the usability engineer did not initially consider.
- 3) As in the Web Components project, changes and fixes are communicated informally through iteration review meetings, or impromptu, face-to-face communications. This is primarily done based on the UI mockups or early demonstrations of implemented code.

After the features for an iteration were delivered, bugs were tracked through the work management system.

Evaluation: By the end of the project, the general steps outlined above allowed QA to work more effectively within the team. Everyone had a better overall understanding of each other's roles and responsibilities. For example, the QA person would make sure to verify any proposed bug fixes that related to the UI with the usability engineer before sending it to the developer. Having the QA person and usability engineer work more closely together in this project proved to be very beneficial for each party as described in point 2 above.

That said, even by the end of the study period, QA, developers, and the usability engineer had some difficulties working together synchronously. This was partially because of the general inexperience of the team with agile methods. Specifically, feature development estimates were not very accurate. Especially in the beginning, this led to iteration plans that specified more features than the developers were able to complete. Given a longer study period, this problem may have subsided.

Another issue that was not addressed was how to work other QA tasks into the development process. One developer noted that the overall reliability of the product was not a primary focus of the project early on. He found that:

“Some things we took for granted such as this app needs to run 24/7 but we've been seeing crashes. So now we've been spending effort on those things.”

In addition, things such as performance testing and concurrency tests were not considered during the study period—things which will need to be taken account as the XSBD process is used to develop other products.

Issue J. Team did not address how to integrate documentation development into XSBD process.

Diagnosis. At Meridium the documentation serves as an important piece of the user experience for their products. In addition, documentation personnel are important in that they have exposure to a broad range of the different products, which gives them insights into things such as UI standards and general functional consistency issues. The documentation person assigned to the Touch Screen project could only commit a limited amount of time to the team due to his involvement in other projects. The documentation person only sporadically attended daily meetings and iteration review meetings. He was thus not able to maintain a good understanding of the project and its progress.

Action. The product manager determined that there would be no separate help system for the Touch Screen project. This was partially because the system itself was relatively simple and easy to use but also because the team was unable to get a larger time commitment from the documentation person.

Evaluation. The team was unable to effectively define how documentation development could work within an XSBD process. As noted in the retrospective at the end of the release:

9/17/08 Retrospective

...

What did not go well:

- Interface changed to the very end so documentation was unable to start their work. (How can documentation be integrated with constantly changing UI. Or can we freeze the UI at some point?)
- ...

Although this did not have much impact on this team in particular since this project required minimal documentation, it will become an issue if the process is used to develop larger software products. Documentation development as it currently stands has the potential to slow the progress of an XSBD team at Meridium since documentation is not started until the UI for the entire interface is frozen. This is a problem in XSBD and similar agile processes since UI development proceeds incrementally and can change throughout the development process. These issues were identified in the retrospective held at the end of the release.

6.4.3 Discussion of Touch Screen project

In using XSBD in the Touch Screen project, we encountered a number of the same issues related to communication, collaboration and information sharing that were encountered in the Web Components project. In some ways, we were able to improve on how XSBD was used compared to that earlier project. This team, which was collocated and in some ways less experienced with agile methods than the Web Components team, experienced a number of different issues. Below, we summarize some of the key findings from using the XSBD approach to develop the touch screen system.

- Define high-level goals and revisit them throughout the project (See Issue A). As in the Web Components project, we found that having a common overall vision of the project was important for the team to develop a product that met customer requirements. Having and maintaining prioritized high-level goals that are agreed upon by the entire team helped to

mitigate the problem of conflicting opinions between team members. However, it is important to revisit these goals continuously and revise them as requirements change and new requirements emerge.

- Have an “Iteration -1” before the project starts if the team is unfamiliar with the development process. Use this time to introduce the process to the team and define each team member’s role. Although this initial training time was helpful, the team itself did not generally accept and understand it well until several iterations into the project (See Issue B). One way to improve this training period may have been to run several tutorial sessions with timed activities that simulate the XSBD approach. This sort of tutorial was later developed and run at ACHI 2009 [65].
- Use of the CDR combined with lightweight usability evaluation techniques can help integrate a scenario-based usability engineering process within an agile development framework. The CDR makes explicit the link between design goals, design decisions and usability testing results and helped the usability engineer develop a more cohesive user interface while designing it incrementally. Claims-based user testing combined with lightweight evaluation techniques allowed the usability engineer to run targeted evaluations more efficiently and deliver value to the rest of the team in a timely manner (See Issue G). In addition, claims were found to be an effective way for storing alternate design proposals for specific features—helping to resolve design disagreements between team members (See Issue F).
- The CDR and the process the usability engineer used to maintain it was itself important in communicating and demonstrating the value of usability to other team members. Although the team referred mostly to the mockups, the usability engineer regularly communicated other aspects of the CDR such as the prioritized goals, design claims and user testing results throughout the project when referring to her work (See Issue D). This helped those who were initially unfamiliar with what usability engineering was and what it was used for.
- A combination of enforced synchronization points between iterations and opportunistic synchronization points within iterations helped keep development and usability tracks working together effectively and helped to prevent drifting between the UI design and the implementation (See Issue E, Issue F). This, combined with a unified planning process that

encompassed both development and usability tasks, also helped the team remain flexible and able to address issues related to design and development problems and changing requirements.

- It is important to have an end user representative on an XSBD team to support the usability engineer and help ensure that the system will meet end user needs. The usability engineer is needed to help interpret requirements, contextual issues and end user needs and translate them into a design that the developers can work with (Issue H). To do this, they need access to that information through the client company. In the Touch Screen project, a single person was able to act as both the customer representative and end user representative since he was currently leading the initiative at the client company and had previous experience working in the site where the touch screen would be used.

At the end of the study period, there were still several open issues that were not completely addressed. They are summarized below:

- The usability engineer was not able to run any large scale, summative usability evaluations at the end of the release cycle (See Issue G). Thus, it is not clear how well such evaluations can be run within an agile team without limiting development velocity. Nor is it clear what benefit such evaluations would be to an agile team that needs to continuously develop new functionality.
- The team did not completely address how to integrate other software development areas—specifically quality assurance and documentation—into an XSBD team (Issue I, Issue J). More work is needed to define how work practices specific to these areas can interoperate with specific XSBD practices and artifacts such as the CDR and parallel development and usability tracks.
- More work is needed to study how XSBD teams will work within a broader software development organization (See Issue C). Meridium, like many other companies, has a number of interdependent teams working on different products. Specifically, how can an XSBD team deal with external dependencies from these other teams or from other areas of the company such as upper management or marketing.

6.5 Overall discussion

This section will summarize some of the key benefits and limitations of the XSBD approach based on the two action research case studies conducted at Meridium. These findings are organized by how the modifications to the XSBD approach and the observed results of those changes addressed the key integration challenges introduced in Section 1.1. Future research efforts based on these case study results are discussed in Chapter 8.

***Balance of power.** In developing a specific instantiation of a combined approach, how should the approach be balanced between usability and agile methods? For instance, an agile-centric approach may involve providing some level of usability-training to agile developers while a usability-centric approach may have usability engineers complete UI designs before agile developers begin their work. The integration approach will have an impact on the long-term viability and acceptance of an integrated process in practice.*

In many commercial organizations like Meridium, it is more practical to take an agile-centric integration approach (See Section 6.3.2: Issue 3). Agile approaches have been widely adopted in industry, and agile organizations may be resistant to certain usability practices which are viewed as ‘non-agile’ [90]. Some agile practitioners have advocated an approach where agile developers are given some usability training [83, 102]. However, usability is a distinct discipline and many UI-focused systems require the expertise of a dedicated usability practitioner. The case studies above have shown how a usability engineer can be effectively integrated into an agile team using the XSBD approach and how a number of different integration challenges can be addressed.

In using XSBD in practice, we found that scenario-based usability engineering can work effectively within an agile development framework. The streamlined CDR, allowed the usability engineer to focus usability evaluations on critical design features through claims linked to high-level design goals (See Section 6.4.2: Issue B, Issue D). By using lightweight evaluation techniques, she was able to run those tests within a single iteration and provide actionable guidance to developers in a timely manner to continuously improve the UI without sacrificing team velocity. Even though the integration approach was agile-centric, changes were not limited to the usability side of things. Certain changes to established agile practices had to be made such as an abbreviated requirements analysis process and the addition of an end-user representative (See Section 6.4.2: Issue H).

At Meridium, this was a practical way to approach the integration of usability into an agile framework. The company was already adopting agile practices, but the people there had little to no experience working with a usability engineer. Thus, they had limited knowledge of usability requirements,

practices, and outputs. By showing, through the CDR, how high-level design goals were linked to specific features that were validated through tests, the usability engineer was able to demonstrate her value to the rest of the team (See Section 6.3.2: Issue 1, Section 6.4.2: Issue B). In fact, this was likely facilitated by the fact that the team was working in an incremental agile development cycle. She was able to show off the results of her work early and often.

***Checks and balances.** How can appropriate tradeoffs between the agile and usability practices in the integrated approach be made? Any integrated agile usability approach will have tradeoffs that may, for example, sacrifice development velocity for validation of the usability of a user interface. The integrated approach should help agile usability teams make these tradeoffs given the specific needs of their development project.*

Prioritized goals defined at the start of the project, agreed upon by all team members and followed through to project completion are important to ensure balanced design decisions that take into account development, usability, and other needs (See Section 6.3.2: Issue 2). These goals should not be limited to usability design goals, but should also include customer, development and company-centric goals (See Section 6.4.2: Issue A). These goals need to be revisited on a regular basis through a unified planning and assessment process as the product is developed incrementally to ensure that goals are being met and that adjustments to the goals are made as requirements change.

The use of prioritized design goals is a central feature of the CDR. The usability engineer was able to directly test whether goals were being met through metrics she defined with the team and give the developers information about the relative importance of changes that needed to be made and how severe problems were (See Section 6.4.2: Issue B, Issue D). Developers were then able to estimate the time needed to implement those changes, giving the team enough information to prioritize and place changes on the backlog.

Although team members had specialized roles in the projects (e.g. usability engineer, developer, QA tester, etc), following the XSBD approach gave team members a broad but shallow understanding of what the rest of the team members were doing (See Section 6.4.2: Issue B). This was important in that it gave each team member a better understanding of the different roles and the importance of each. This was especially important for the Meridium employees, as they did not initially have knowledge of usability practices.

The Touch Screen project team members made an effort to define and follow prioritized goals, but they were not completely successful in defining goals that covered a broad range of concerns (See

Section 6.4.2: Issue A). For example, the need for defining a software reliability goal was not identified until late in the project. This may have been a result of the relatively short development period of the case study. In addition, product and company-wide goals were not defined due to difficulties in working with and coordinating with other groups and upper management (See Section 6.3.2: Issue 6, Issue 7; Section 6.4.2: Issue I, Issue J).

Synchronicity. *How can the agile development and usability engineering sides of a project stay synchronized throughout the development effort? This is critical as design drift between the interface design and its implementation can render the usability work irrelevant and result in a system that has poor usability. This problem is exacerbated by the speed at which agile development projects proceed. Addressing this issue depends on proper development and use of shared design representations, communication practices and process planning structures within the team.*

An integration approach that relies on parallel usability and development efforts requires a well-defined process framework in order to work effectively. The XSBD approach synchronizes development and usability through regular checkpoints at the start and end of each iteration. As one iteration ends and another begins, specific handoffs between the developers and usability engineer need to occur (See Section 6.3.2: Issue 4, Section 6.4.2: Issue F). The other, equally important (if not more so) feature is regular, close communication within each iteration between the two groups (See Section 6.4.2: Issue F). These nonspecific, opportunistic meetings are necessary to ensure that implementations are matching the UI designs and to address issues as they arise. This allows for the flexibility in the process framework that is necessary to account for the constant change and uncertainty that surrounds the development process (See Section 6.4.2: Issue E).

The XSBD approach defines a specific strategy whereby usability engineers and developers collaborate in order to effectively work in parallel (making it similar to other approaches, like that defined by Miller and Sy [84, 127]). It makes clear what is being delivered to what person and when. These handoffs occur at regular, structured meetings at the start and end of each iteration. However, despite this framework, both case studies highlighted the central importance of communication within agile teams (See Section 6.4.2: Issue D). Regular communication proved to be more difficult in the distributed Web Components project, as the team had to rely more on asynchronous communication methods (See Section: 6.3.2: Issue 3, Issue 4). In addition, interpersonal relationships between team members introduced difficulties in how well development and usability tracks maintained synchronicity (See Section 6.4.2: Issue F). Specific social issues such as this need to be addressed given the importance of communication in agile methods. Although these were touched upon in the Touch Screen case study, they were not a focus of this research and remain an area for further study.

6.6 Chapter summary

This chapter described how the utility of the XSBD approach has been substantiated in practice using an action research case study approach. Action research supports the study of complex social processes through a process of continuously introducing changes and observing the effects of those changes in action. This made it an ideal approach to both study the XSBD approach in the context of a real world development project while simultaneously improving the process in an iterative fashion.

I partnered with Meridium, Inc., a software and services company, and had two development teams use the XSBD approach. Through these case studies, I studied three of the key integration challenges introduced in Chapter 1. Specifically, I explained how the XSBD approach was developed to be agile-centric—meaning that usability practices were adjusted to fit within an agile incremental development cycle. I also detailed how design decisions are made through tradeoffs that balance usability and development concerns by using the CDR. Finally, I described how agile and usability efforts within a project remain synchronized by using well defined coordination and planning mechanisms.

The next chapter will consist of an analytic evaluation of the XSBD approach that resulted from the action research case studies described in this chapter. It will describe how XSBD aligns with key agile and usability values and how it addresses the key divergence points between the two approaches.

7 Principle-based analytic evaluation of XSBD

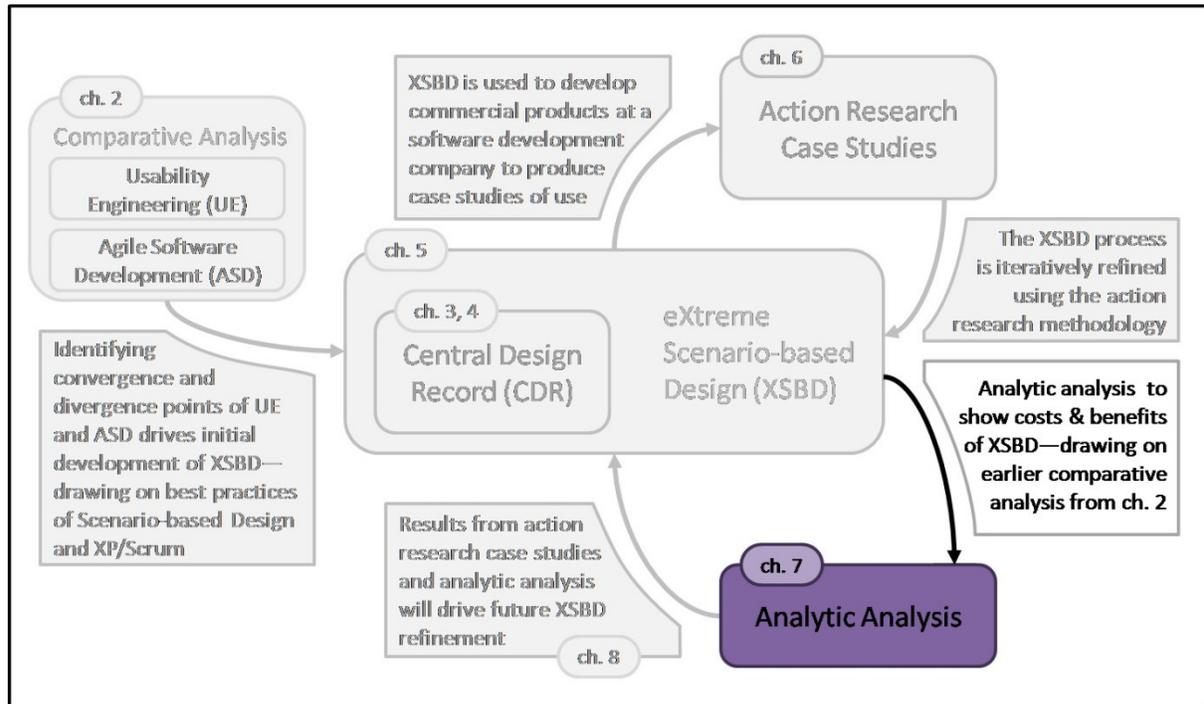


Figure 39. Overview of solution approach. Chapter 7 will include an analytic analysis of the XSBD approach with respect to key agile and usability values/principles.

This chapter will include an analytic analysis of how the XSBD approach addresses the first issue presented in section 1.1: *Challenges facing an integrated approach* (See Figure 39):

1. **Comparison of core principles.** In considering usability methods and agile methods, what are the convergence and divergence points from a high-level principles perspective? It is important to identify where difficulties lie in integrating the two approaches, where similarities can be exploited in a combined approach and what the costs and benefits of the resulting approach are.

I will first show how the XSBD approach—which takes the general approach of integrating usability practices into an agile development framework—is an agile process by showing how it adheres to the core agile values detailed in the Agile Manifesto [13]. I will then demonstrate how XSBD is still a valid usability engineering approach by detailing how it still follows key principles and practices of usability engineering processes.

Next, I will detail how the XSBD approach successfully integrated usability into an agile framework by leveraging key convergence points between the two areas and addressing how divergence points were addressed—highlighting key benefits and costs of that specific integration approach.

Based on my analysis, I will provide a modified set of principles modeled on the Agile Manifesto that any agile usability process should follow. This set of principles will be drawn primarily from the results from my work in XSBD. Connections and similarities to other agile usability approaches will be highlighted to help demonstrate the broad applicability of the values and principles.

7.1 Evaluating whether XSBD is an agile process

Agility is not defined by any one or set of specific practices. Certainly, practices such as incremental development, on-site customer collaboration and test driven development are common among practicing agile teams and specific methodologies. However, common ground among agile methods was established through the Agile Manifesto which captures the core values that underlie all agile methods [13]. Thus, in determining whether XSBD can be said to be an agile process, one should evaluate whether it adheres to the core values as stated in the manifesto.

The Agile Manifesto is reproduced below:

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

These values highlight the central importance of adaptiveness and responsiveness to change that the agile practitioners felt were the most important features of agility [13, 36]. A common misconception among developers unfamiliar with agile is that in agile methods practitioners do no planning and generate no documentation [3]. However, agile teams do have to do those things to work effectively with varying levels of intensity as was experienced in the two XSBD project teams at Meridium. Rather, it is important for agile methods to focus on the items on the left of each of the four values in order to ensure that a product is efficiently developed that meets customer needs. The four sub-sections 7.1.1 to 7.1.4 will describe in detail how the XSBD process is in alignment with each of the core values.

7.1.1 Individuals and interactions over processes and tools

Agile methods stress the central importance of people in development process and how well they communicate with each other. The abilities and dedication of the agile team members will always trump any particular process or tool that is used. This is in contrast to many past software development efforts where people are treated as resources that can be compared interchangeably with time [19]. For example, XP, one of more popular agile methods, has a number of practices that depend on this principle. The customer is working with the developers through the entire project and should be available at all times to answer questions [12]. In addition, pair-programming is used as a way for developers to collectively become familiar with the whole system and enhance team-wide communication. The following practices from XSBD illustrate how it adheres to the agile principle of individuals and interactions over processes and tools.

- In XSBD, there is a requirement not only for continuous contact with the customer, but also continuous contact with an end user representative (See Section 5.2). The end user representative should be someone knowledgeable about the domain in which the system being designed will be used and be familiar with the tasks it will support. This person should be in continuous contact with the team—especially the usability engineer—to answer questions and to help coordinate site visits and evaluations with end users.
- One focus of XSBD is to increase collaborations and communications with end users (See Section 5.2). Practices such as on-site visits, observations, interviews and usability evaluations, will help the usability engineer to better understand the users and the context in which the system is used which in turn will help her develop a more usable end product.
- XSBD stresses the importance of communication and collaboration among team members, particularly between developers and the usability engineers. Proper synchronization of the parallel development and usability tracks depends on it (See Section 5.4.2). This is done through a combination of enforced and opportunistic synchronization points in the form of meetings, and impromptu communications. It is also supported through the CDR, which helps teams to make balanced design decisions by taking into account different, often competing design goals.
- As demonstrated in the two case studies, one of the main focus points was on how best to leverage and adapt the practices of XSBD to fit the teams using them (See Chapter 6

Introduction). Using the action research approach and project retrospectives, we were able to continuously reflect on and improve the process based on the circumstances and needs of each project. For example, the CDR was streamlined to focus on the link between high-level design goals and design features validated through user testing. This helped the usability engineer work incrementally and better communicate usability requirements and designs to other team members (See Section 6.4.2: Issue B).

7.1.2 Working software over comprehensive documentation

Agile methods value working software over comprehensive documentation because the software as the primary measure of process because the delivered system is the ultimate goal of the development process [36]. Agile practitioners and other software developers have noted that documentation can easily become difficult to understand, out of date and can take a significant amount of time to maintain. Although XSBD requires additional documentation related to the design and evaluation of the system, the focus is always on delivering a functioning, usable end product. Specific examples illustrating this focus are presented below.

- The primary goal of using the XSBD process is to efficiently develop a usable software system (See Section 5.1.1). Thus, for an XSBD team, the primary measure of progress is not just working software, but working software that meets high-level project goals. The design documentation, including specific components of the CDR such as the high-level design goals, scenarios and claims, are necessary and needed only insofar as it helps the usability engineer and the agile team meet that primary goal (See Section 5.3, Section 6.4.2: Issue D). This is made clear in the action research case studies, where the approach (and documentation generated and managed through it) were continuously adapted to fit the needs of the teams using them (See Section 6.1.1).
- Efforts were continuously made to streamline and optimize the documentation generated through the XSBD process to better serve the team. For example, the CDR was simplified by reducing the claims map to the core conceptual idea of linking specific goals to design features validated through usability test results (See Section 5.3.1). We also worked continuously to determine how best to communicate usability designs and data to the rest of the team. Often times, this was through face-to-face communication and supported by mockups and scenarios (See Section 6.4.2: Issue D). For the distributed Web Components project, this was done through more documentation asynchronously accessed through a central repository. (See Section 6.3.2: Issue 5).

7.1.3 Customer collaboration over contract negotiation

Agile methods value customer collaboration over contract negotiation to stress the central importance of customer involvement in the development project. Also implicit in this statement is that simple negotiating a contract at the beginning of the project is insufficient. Within the incremental agile development process, continuous collaboration is needed. Customers are needed to define and refine requirements as they review developed functionality and answer questions and concerns of developers as they come up. The XSBD process adheres to and extends this value to include not only customer collaboration but also end user involvement through things such as site visits, interviews and usability testing throughout the process. Specific examples of this are presented below:

- XSBD was a defined customer representative role (See Section 5.2) that is similar to the customer role in methods like XP. Optimally, the customer representative will be from the client company and will work onsite with the team throughout the development project. However, other people can take this role if necessary.
- In the Web Components project, a product manager at Meridium acted as the customer representative (See Section 6.3.1). This was because the project was not being developed for any one specific customer. He was working onsite with the team throughout the project to define requirements and review functionality.
- In the Touch Screen project, the customer representative was an employee at the client company who worked with the team remotely. The team met with the customer representative remotely on a weekly basis and he was able to help define requirements and provide regular feedback on the system as it was developed (See Section 6.4.1). At times, the customer was unsure of what they wanted or what the best course of action was. The team worked closely with the customer to develop a system that would meet the client company's needs best—even when that resulted in disagreements between the customer and some of the other team members. For example, even though the customer initially wanted to include a physical keyboard with the touch screen for data entry, the usability engineer was able to convince him not to include it by showing user testing results that demonstrated that an on-screen keyboard would work best (See Section 6.4.2: Issue H). This demonstrates how closely the customer and other team members worked together and the trust that was developed over the course of the project.

7.1.4 Responding to change over following a plan

Agile methods accept the fact that requirements and project circumstances will continuously change during the project. Rather than rely on precisely constructed plans and try to adhere to them, agile methods focus on adaptability and the ability to respond to changing requirements and circumstances [36, 69]. This means using flexible, short plans and continuously prioritizing and reviewing requirements and the system is developed. Much of XSBD was developed with this value in mind and the following examples show:

- XSBD, like all agile methods, is a cyclical process. These development cycles are centered on the CDR. The user interface is incrementally designed by the usability engineer and is validated by regularly running usability evaluations to verify that the design is sound and is meeting the high-level design goals (See Section 5.4.1).
- In XSBD, detailed planning is generally done one iteration at a time. Similar to methods like XP and Scrum, project team members are assigned tasks based on the relative importance of features that need to be developed and evaluated. Release planning is also done which may comprise 4-8 iterations, but these plans are done at a high level and are subject to change. Even within iterations, the plan may change depending on project circumstances. For example, in the Touch Screen project, the developers were unable to deliver the features at the end of the first iteration and had to push some of them into the next iteration. The usability engineer, in turn, had to change her usability testing plan on the fly since she would not have the working system to use in the evaluation (See Section 6.4.2: Issue E). XSBD teams, and the planning process they used, were flexible enough to handle situations like this.
- In instances where the XSBD project was used, the focus was always on continuously reviewing and adapting the process to best fit the needs of the team. Using the action research approach and project retrospectives, both the Web Components team and Touch Screen team were able to adapt the project to their individual needs. Indeed, any team that uses the XSBD approach will need to tailor it to fit their specific project and work context.

7.2 Evaluating whether XSBD is a usability engineering process

Unlike agile methods, usability engineering does not have the benefit of a single, universally accepted manifesto from which all methods can draw from. However, usability engineering processes generally

do share a common set of core characteristics that I will use to show that XSBD is a usability engineering process.

Usability is a quality attribute that evaluates how easy user interfaces are to use. As specified by Nielsen, usability is composed of five components: learnability, efficiency, memorability, errors and satisfaction. The goal of usability engineering is ensure that the system being designed has the optimal usability for the end users [111]. Implicit in these definitions is the central focus placed on the end users in developing a system. This is evident in the three core characteristics common to usability engineering processes [35, 38, 54, 95, 107, 111]:

- **Requirements analysis:** Gaining an understanding of the context of use, the end users, and what the users need to do. Usability goals are defined based on this analysis.
- **Design:** Developing a design solution towards meeting those usability goals.
- **Evaluation:** Evaluating the design to verify that the design meets those goals.

Usability engineering processes operate in an iterative fashion so that the three steps defined above form a cycle from problem analysis to design to evaluation and back. The output of each step is dynamic and subject to change based on the output from the previous step. Thus, the usability goals and requirements might change based on how users reacted to the design [28]. This reflects the importance of responding to change and dealing with uncertainty, one of the main similarities between agile and usability processes. However unlike agile methods, usability methods develop their designs in layers rather than slices. That is, there is generally an up-front requirements analysis phase followed by iterative phases where large portions of the UI are designed at increasing levels of fidelity [38, 111]. Practices such as paper prototyping, low-fidelity prototyping, Wizard-of-Oz testing allow usability engineers to design and evaluate the UI quickly early on in the development effort without investing in significant coding effort.

This section will summarize how the XSBD process embraces the three characteristics of usability engineering processes and follows them in an iterative fashion to meet end user goals. It will specifically focusing on how it adheres to core practices from scenario-based usability engineering as defined by Carroll and Rosson [111].

7.2.1 Requirements analysis

Usability processes like SBD rely on a requirements analysis phase to define project goals, gain an understanding of the end users, their work context and their activities [111]. They acknowledge that

all requirements cannot be defined at the beginning of the process. However, this analysis is generally done in phases, with a large scale requirements analysis phase at the beginning of a project. This section summarizes how the XSBD process maintains the major parts of this requirements phase.

Defining project goals. Having some type of project vision, which contains a shared understanding of the project goals, is important to define at the beginning of the project. In standard SBD, this consists of things such as a statement of the project vision and rationale and an initial analysis of project stakeholders. In XSBD, project visioning is an important part of the start of any project (See Section 5.3.2). A description of the vision of the system, user descriptions and prioritized design goals are developed and agreed upon by all team members at the start of the project. The prioritized design goals becomes a critical piece of the CDR, and is subsequently used by the usability engineer as she incrementally develops the user interface to verify that the system is adhering to the overall vision. As shown in the Touch Screen project case study the usability engineer is able to define metrics against these goals and run usability evaluations in each iteration to demonstrate how the system was meeting them (See 6.4.2: Issue B). Doing so was important in that this helped the rest of the team understand what the usability engineer was doing and demonstrated her value to the rest of the team

Understanding users. Many usability engineering processes are ‘user-centered’, in that they stress the importance of developing a detailed understanding of the end users in order to develop it so that it best meets their needs [111]. This includes understanding their needs, limitations and preferences in the context of their work. User involvement is critical to getting this understanding. SBD does not require any specific set of techniques but introduces a number of different ways this can be done. For example, contextual inquiry—in which people are actively observed and questioned as they complete their day to day tasks—can be used to better understand user intentions and thoughts [14]. In fact, Holtzblatt and Beyer have developed rapid contextual inquiry specifically to work within an agile context. Many agile teams have adopted personas—which are written documents that consist of narrative descriptions of hypothetical user’s behaviors and activities—as proxy users that developers design their systems against [97].

In XSBD, understanding users and their motivations is a central concern. Understanding users, their motivations and characteristics is an important part of project visioning (See Section 5.3.2). In addition, XSBD has the concept of an ‘end user representative’, preferably a potential end user of the system who works with the team to answer questions about their preferences, work activities and to get their feedback about the system being designed (See Section 5.2). In the Touch Screen project, the

same person from the client company was able to act as both the customer representative and the end user representative because of his past experience with the work tasks the system was meant to support (See Section 6.4.1). In the Web Components project, the product manager acted as the end user representative due to his knowledge of and past experiences with the client companies (See Section 6.3.1).

Understanding activities and work context. In order to develop the system to best meet user needs, designers need to understand the context in which the system will be used. This means understanding what the end users need to accomplish and the circumstances and environment in which they work. In SBD, a number of different approaches are suggested to gain this understanding including workplace observations, recordings of users at work, interviews and collecting artifacts from their workplace. In XSBD, any of these can be used as no specific set of approaches are required. For example, in the Touch Screen project, the usability engineer was able to conduct a site visit where she was able to observe the area where the system would be installed. In addition, she interviewed project stakeholders including potential end users to better understand how and where they worked (See Section 6.4.2: Issue C). The difference between XSBD and usability approaches like SBD is that these activities are not largely done at the start of a project [38, 111]. The requirements analysis iteration at the start of the project means that certain things such as extensive field studies cannot be done all at once and must be interspersed between releases. In addition, less time consuming activities such as end user interviews are better suited to XSBD. Having a less complete understanding of work context is offset by having an end-user representative on the team. In addition, being able to design and deliver working parts of the system more quickly is beneficial in that feedback from end users can be obtained earlier and more often which can benefit the usability of the end product [127].

In SBD, requirements data is synthesized into detailed stakeholder, task and artifact analysis artifacts such as a hierarchical task analysis [111]. These are supported by scenarios which illustrate data gathered about the work context and activities, and by claims which highlight important features of current practice. Problem scenarios and claims are used similarly in XSBD as they are in SBD. Problem scenarios are more useful near the start of the release cycle and helps users to understand users, their goals and their work practices. Claims, however, are used throughout the XSBD process to highlight critical areas of the current system and guide usability evaluations as the system is developed incrementally (See Section 5.3.3).

7.2.2 Design

During design, usability engineers design the user interfaces that meet user needs identified during requirements analysis. This includes designing the activity workflows, the specific interactions to that enable those workflows, and the information that is presented to the user. This section summarizes how the XSBD process supports the above activities, which are referred to as *activity design*, *interaction design* and *information design* in SBD.

Activity Design. In activity design, the goal is to specify system functionality [111]. That is, the team identifies and organizes the operations the user is able to do with the system. Scenarios provide easily understandable and flexible examples of usage in the users work context. In XSBD, scenarios—especially activity scenarios—are used for just this reason. In the Touch Screen project, they were used early on with associated storyboards to explore how the system might function (See Section 6.4.2: Issue E). In the Web Components project, they were less useful since the functionality was largely identical to the existing desktop application. However, in both projects, claims analysis was important in that they allowed the usability engineers to identify critical design features that related to high-level goals and verify their design rationale through usability evaluations.

Information Design/Interaction design. In information design, the usability engineer designs how information is presented to the user. Information design in the context of usability engineering is not about designing striking visual design schemes and icons. Rather, it focuses on presenting information to the user that they are able to easily complete their tasks [111]. Within SBD information design is done within the context of the Gulf of Evaluation. This is a concept developed by Norman that refers to the ‘cognitive distance’ between the information presented to the user by the system and the user’s current goals [99]. The Gulf of Evaluation is ‘closed’ by ensuring that users are able to properly *perceive*, *interpret* and *comprehend* the information so that they are then able to interact with the system.

In interaction design, the usability engineer designs how the user interacts with the system (e.g. inputting data, making selections, changing options, etc) [111]. In SBD, interaction design covers the Gulf of Execution. This refers to the distance between what the user wants to accomplish and the actions he has to take with the system to meet his goal. The Gulf of Execution is ‘closed’ when the user chooses a system goal that corresponds to his user goal (e.g. if his goal is to document that his work is done the *system goal* would be to mark the work order as closed using the system), develop an

action plan which consists of the interaction steps required to complete the goal, and *executing* that plan by using the system.

In XSBD, some activity, information and interaction design is done in each iteration since work is done incrementally. The usability engineer has to completely specify the user interface for one piece of the system in one iteration so that the design can be handed off to the developers to implement in the following iteration (See Section 5.4.1). This ensures that usability engineers and developers can work in parallel to optimize the overall velocity of the team. Gulf of Evaluation and Gulf of Execution concerns are therefore continuously addressed by the usability engineer. As stated previously, most scenarios tended to be activity-oriented as they were useful in describing user actions and goals. They corresponded more to the *system goal* stage of the Gulf of Execution.

In defining how the other stages in the Gulf of Execution and Gulf of Evaluation were addressed in the designs, the usability engineers relied on a combination of artifacts. A number of different prototyping techniques were used to define and communicate designs to other team members. Prototyping is a very important usability engineering technique as it allows her to efficiently explore the design space, refine user requirements and communicate design specifications to other team members. In XSBD, face-to-face communication in combination with annotated storyboards or mockups were commonly used in place of information and interaction scenarios to show both how the system displays information to the user and how the user interacts with it. Relying on these techniques allowed the usability engineer to explore both Gulfs in an efficient manner. Team members in the Touch Screen project were reluctant to read through detailed narratives like those described in information and interaction scenarios and preferred direct communications with the usability engineer as well as visual mockups (See Section 6.4.2: Issue D). Indeed, it is acknowledged in the Rosson and Carroll book on SBD that attempting to write and maintain a large number of scenarios can quickly become cumbersome [111]. In XSBD, claims are used as a lightweight mechanism for documenting design tradeoffs for critical design features. These claims could relate to any combination of activity, information or interaction concerns. For example, in the Web Components project, the usability engineer documented one claim related to a task bar that included tradeoffs that related to how well users might be able to *interpret* and *comprehend* the control (See Section 6.3.2: Issue 4). In the Touch Screen project, the design mockup shown in Figure 34 includes a number of claims that relate to both information and interaction features.

7.2.3 Evaluations

Like agile methods, one key characteristic of usability engineering processes is their cyclical nature. Usability evaluations are any analysis or empirical study of the usability of the system [111]. They provide feedback on whether design is meeting the project requirements, identify usability problems, and can also uncover new requirements.

One important part of the evaluation process in SBD is to define usability specifications. These specifications are precise and testable statements of the usability characteristics required of a system. This is a critical part of XSBD. The usability engineer defines metrics against the high-level design goals which are tested after each iteration to verify that the system is meeting those high-level goals (See Section 5.3.2). This is important in XSBD as it allows the usability engineer to ensure that the overall design is coherent and meeting user needs as it is developed incrementally.

XSBD like other usability engineering methods leverage a variety of analytic and empirical evaluation methods. Analytic methods detailed analyses or theoretical models of the user interface, usually done by usability experts [111]. They are useful in that they are often less expensive and time consuming to run. The identification of claims is itself an analytic evaluation method used in XSBD by the usability engineer to highlight key tradeoffs of the user interface that will need to be more extensively tested later on (See Section 5.3.3). In the Touch Screen project, the usability engineer had access to a developer who had previous experience with touch screen UI development and was able to do expert inspections of the interface as it was being development (See Section 6.4.2: Issue G).

Empirical methods such as controlled usability testing in a laboratory or longitudinal studies of the system in use are better in that the usability engineer will be able to obtain actual feedback from users in realistic situations. One problem with many empirical techniques are that they can be time consuming and costly to run. It is difficult to fit these types of evaluations into an agile framework where teams are expected to design, implement and test a piece of functionality every 2-5 weeks. XSBD makes extensive use of empirical studies but relies more on lightweight or discount usability evaluation methods [93]. These methods are less controlled and rely on fewer subjects per study but can still find a large number of usability problems. Because these types of evaluations require less planning and fewer resources, they can be planned and run easily within a single iteration. In the Touch Screen project, the usability engineer was able to run an evaluation in almost every iteration by using a variety of tactics including recruiting other Meridium employees that act as participants,

employing the ‘Wizard of Oz’ technique to test the interface without the touch screen hardware and think-aloud protocols (See Section 6.4.2: Issue E).

Although most evaluations in the XSBD process are more formative in nature, in principle more extensive, controlled evaluations should be run at the end of each release cycle to provide more comprehensive feedback and to verify the fixes made based on previous evaluations. In practice, neither the Web Components nor the Touch Screen project was able to run this type of ‘summative’ evaluation so it is unclear how well they work within XSBD or whether their benefits offset the costs of running them (See Section 6.4.2: Issue G).

7.3 Addressing the conflicts between usability and agility

Sections 7.1 and 7.2 demonstrated how XSBD adheres to core tenets and characteristics of agility and usability. However, in integrating usability into an agile framework, a number of conflict points between the two areas became apparent. These conflict points were identified during the action research case studies of the XSBD approach at Meridium, the analytic analysis of the approach comparing it to core agile and usability characteristics and through a comparison of XSBD to other agile usability approaches. This section will highlight these specific conflicts, explain how they are mitigated in the XSBD approach, and highlight both the benefits and costs of those mitigation strategies

7.3.1 Incremental development vs phased development

Although both agile and usability methods follow cyclical development processes, they differ in what work goes into each of those cycles and how fast those cycles generally go. Agile methods tend to use incremental development cycles where during each iteration some piece of functionality is designed, implemented and tested. The idea is to have a high quality, ‘deliverable’ system at any point in the development process [11, 12]. This allows customers to give feedback on the system early and often and validate that it does what they want it to do. Usability methods like SBD tend to follow a more ‘layered’ or iterative approach where the requirements are first defined and the system is then completely defined at increasing levels of fidelity [16, 111]. By better understanding the user and the context of use, designers can look at things more broadly and deliver a more cohesive UI design [38]. The problem with using this approach from an agile perspective is that it is viewed as ‘big design up front’ where designers try to completely define the problem and solution space from the start. This can lead to wasted design work and excessive documentation that may become out of date as

requirements change. In addition, if this approach is followed then developers will have to wait for the usability engineers to finish this phase before they can start working [90].

In XSBD, this problem is addressed by integrating the core SBD phases (requirements analysis, activity design, information design, interaction design) into the incremental development framework. This was the practical way to address this conflict as agile methods are widely adopted in industry and agile organizations are more likely to adopt usability practices if they fit within their established development frameworks. This proved to be the case at Meridium. In XSBD, instead of a large scale requirements analysis phase, there is an abbreviated requirements analysis iteration where just enough data is collected and analyzed for the team to begin working. The usability engineer can do additional requirements gathering later on at the start of subsequent iterations as necessary (See Section 5.4.1). This approach is common in other agile usability approaches and proved to be sufficient for the development efforts at Meridium. Within each iteration, the usability engineer works on designing the UI for the part of the system that will be implemented by the developers in the next iteration. By managing and relying on the CDR, the usability engineer can verify that the overall vision of the UI design is maintained by verifying that the designs match the high-level goals of the system (See Section 5.3). Working within this incremental framework also allows the usability engineer to work in parallel to the agile developers, an approach popularized by Sy and Millar, which ensures that neither side is left idling [85, 127].

The usability engineer for the Touch Screen project noted that although the overall design was not as cohesive as it might have been if she had designed it all at once, it appeared to be more usable for the end users (See Section 6.4.2: Issue G). This was because she got feedback from users earlier and more often which she was able to incorporate into her designs. It is not clear how well this approach would work for larger, more complex systems. Those types of projects might need a longer requirements analysis iteration or longer iterations in general.

The tradeoffs of this approach are summarized below:

Designing the user interface within an agile incremental framework

- + Can result in more usable design due to early and more frequent feedback from users
- + Maximizes team velocity by allowing usability to work in parallel to development
- May result in less cohesive overall design
- Unclear how well this approach will scale with project size

7.3.2 Working software vs design documentation

One of the foundations of the Agile Manifesto is that working software is valued over comprehensive documentation [13]. Past software development projects would often get bogged down by large requirements and design specification documents that were difficult to maintain and would quickly become out of date. In agile methods, high quality working software is valued above all else since that is what is being delivered to customers [36]. Thus, agile methods like XP strive to minimize documentation to only what is absolutely necessary through practices like onsite customers and close collaboration between team members [36]. Usability engineering methods like scenario-based design would appear to work against this principle as they use a variety of different design artifacts to develop the UI interface before any code is written. In SBD, a variety of different types of scenarios are developed to describe current work practices and the system being developed. In addition, a variety of low and high-fidelity prototyping techniques such as sketches, storyboards and click-through mockups, are used to design and evaluate the interface before it is implemented.

XSBD still adheres to the core agile concept that high quality software should be valued most. Any practices and design artifacts should be developed and maintained only if they help the team meet that goal. The difference here is that implicit in the ‘high quality’ statement for agile usability teams is that the software should be highly usable for the end users. This means only documenting what is necessary for both the usability engineer and agile developers to work together effecting in efficiently developing a usable system. For example, the claims map (See Section 4.2.4) was an interconnected map of design claims show how specific features mapped to activities which were in turn linked to high-level goals of the system. This representation was criticized by agile practitioners for being too difficult to maintain and would be of limited value to other team members [64]. This representation was simplified to the core concept of linking high-level goals to design claims which are validated through usability tests. This proved to be more useful and easier to understand by other team members in the Touch Screen project while still helping the usability engineer plan evaluations and ensure that the UI design was meeting high-level goals (See Section 6.4.2: Issue B). In addition, UI prototypes the usability engineer developed were ‘mid-fidelity’ and lacked things such as specific non-standard workflows and error conditions (See Section 6.4.2: Issue F). These prototypes proved to be adequate in that the usability engineer could communicate design intentions would not spend too much time developing them. She could then communicate holes in the UI design to other team members as necessary through face-to-face communications. Finally, in XSBD only activity scenarios are typically developed to give the team a basic understanding of the system workflow early on in the

development effort. Information and interaction scenarios were not used as specific details about the UI could be communicated through the UI mockups and face-to-face communications.

However, in spite of some of the streamlining efforts to SBD design artifacts designed above, some team members in the Touch Screen project still felt that some of them were unnecessary and extraneous (See Section 6.4.2: Issue D). As the other team members were unfamiliar with working with a usability engineer, they were initially unsure of the type of activities she did and why. They were also in turn unfamiliar with some of the design artifacts she would produce such as interview questionnaires, user testing protocols and design claims. This disagreement in terms of what is or is not needed in terms of documentation was therefore an artifact of role specialization within the team (See Section 7.3.5).

The tradeoffs of this approach are summarized below:

Streamlined design documentation within XSBD supported by close communication with other team members

- + Allows usability engineer to focus on delivering usable designs and minimize unnecessary documentation work
- + Aligns with agile framework through focus on developing highly usable working software
- Specialized roles within team may cause disagreements in terms of what constitutes needed or extraneous documentation.

7.3.3 Customer focus vs end user focus

Both usability and agile approaches are ‘human-centered’ in that they value close collaboration with stakeholders. However, they differ in which stakeholders they focus on most. One of the core concepts of agile development is close collaboration and continuous with customers. In XP, there is an on-site customer who joins the team and works with them throughout the development project to define requirements, do acceptance testing and answer questions as they arise [12]. In agile teams, the ultimate goal is to efficiently develop a high quality product that meets the user’s needs. Usability engineering methods like SBD are user-centered, meaning that the ultimate goal is to develop a high quality system that meets the end users’ needs. Usability engineers use a variety of user-focused techniques such as onsite observations, interviews, participatory design and user testing to understand users and ensure that the system meets their needs [111]. For many development projects, the customer and client company will not be the ultimate end users of the developed system. This can

result in conflict between the usability engineer and the agile developers because of their focus on different stakeholders.

In XSBD, like in other agile methods, the ultimate goal of the team is to develop a high quality system that meets the customer's needs. In practice, since the customer is the entity that is funding the development of the product, ultimately he has final say on what is being developed and why. This reflects the 'agile-centric' I took in developing XSBD and instantiating it in practice and relates back to one of the core challenges facing an integrated agile usability approach (See Section 1.1). The key difference between XSBD and a standard agile approach like XP or Scrum is that usability is one of the quality attributes of the system. This means part of ensuring that the system meets the customer's needs will be to ensure that it meets the needs of the end users.

This conflict is partly addressed by collectively defining and prioritizing the high-level goals of the system. Doing so during the visioning of the system will help to highlight the relative importance of usability with respect to the customer's other goals. This is described in more detail in Chapter 5 (Section 5.3.2) and in section 7.3.6. However, In XSBD this conflict is directly addressed by having an end user representative on the team. The end user representative is the usability analogue to the customer representative. His role is to provide the usability engineer with contextual information by answering questions, periodically evaluating the system and acting as a liaison to help recruit end users for usability testing (See Section 5.2). This is a distinct role in XSBD because the customer representative will not have access to this type of user-centric information—especially if he will not be the one using the system. In addition, this helps to highlight user-centeredness as one of the key drivers of XSBD and highlights the different needs of the usability engineer.

Having a distinct end user representative role on the team can act as a counter-balance to the customer representative and help ensure that usability concerns are properly being considered by the rest of the XSBD team. However, there are additional costs required to bring another person onto the team. In practice, the same person can fit both the customer role and the end user role as was the case in the Touch Screen project (See Section 6.3.1). As a manager at the client company, this person had a good understanding of the project scope and overall functional requirements. In addition, as the user representative, he had detailed knowledge of the environment and the people that would be using the system. Finding a person that can fill both roles may be difficult depending on the client company, the product being developed and other factors.

The tradeoffs of this approach are summarized below:

Having a distinct end user representative role on the XSBD team

- + Can act as counterbalance to customer representative role, ensuring that usability concerns are continually addressed
- + Can support the usability engineer by acting as liaison to other end users and their work environment
- Added cost of bringing another person onto the team or finding someone from client company that can fill both end user role and customer role

7.3.4 Test driven development vs usability test driven development

Test driven development is one of the most common agile development practices. Agile developers continuously create automated unit tests that define what the software is supposed to do before writing the code itself [12]. This practice allows developers to incrementally develop the system while ensuring that the code base remains robust even as it is evolved and refactored. It also allows developers to identify design flaws sooner, discover problems in the requirements and diagnose and fix problems in the code more quickly. In addition, these test suites can be run automatically on a daily basis. There has been some work tools that can automate the usability testing process. However, such tools depend on developing accurate models of human behavior which in itself is an active research area [55]. By and large, usability testing does not have the benefit of the level of automation that test driven development supports as they require human intervention both in the sense that they rely on feedback from actual users and they often depend on a usability expert to analyze and interpret that feedback. As a result, usability processes tend to take a less nimble approach than agile methods by focusing more on gathering up-front data beforehand and doing early lightweight prototyping iterations before code development begins. For example, the agile concept of refactoring does not have a clear analog in usability approaches since the impact of making small changes to the user interface can often not be verified until after the next set of usability evaluations are run with users.

Conceptually, usability testing is more similar to acceptance testing. In agile methods like XP, acceptance testing is where the customer verifies that the system functions as he specified [12]. Usability testing is similar except that it is done to ensure that the system is usable for the end users. In XSBD, certain practices are put into place that allow the usability engineer to design an effective UI while still maintaining some of the flexibility and development velocity needed in an agile team. Unlike agile developers who leverage automated test suites, the usability engineer cannot easily run an evaluation of the entire UI at the end of each day. In XSBD, the usability engineer documents claims

as she designs each increment of the UI to highlight specific UI tradeoffs that relate to the high-level goals of the system. These identify specific features to evaluate in the next round of evaluations after the functioning code is delivered. This is conceptually similar to the test driven development process where tests are written before coding begins. In conjunction with lightweight evaluation techniques, the usability engineer can then evaluate the functionality delivered in the previous iteration and have provide redesign suggestions by the start of the following iteration. And since the redesign suggestions can be traced back to high-level goals through the CDR, the team can have a better understanding o the relative importance of those changes.

This continuous, incremental testing approach can be beneficial in that actual feedback from users is obtained earlier and more often which can help identify problems and new opportunities. In addition, because agile methodologies have been developed to reduce the cost of making changes to functionality throughout the development project, these more frequent UI changes will not have the same additional development cost as if they were made in a team using a more traditional development approach. However, in practice, these tests do result in additional churn in the system design which results in additional work for the developers. Constant changes in the UI can become frustrating for them as was documented in Touch Screen project (See Section 6.4.2: Issue F). These problems can be minimized by having the usability engineer and software developers communicate and collaborate continuously within each iteration even as they each work on their own specific work tasks. In addition, freezing the UI design at the start of an iteration and moving significant changes based on UI evaluations to the product backlog to be reevaluated for the following iteration can minimize developer frustrations.

The tradeoffs of this approach are summarized below:

Using small-scale continuous UI testing to drive development efforts

- + Encourages flexibility and openness to change in the UI, which is in alignment with agility
- + Can get feedback from users earlier and more often, which can help ensure the UI is meeting their needs
- Increases work for agile developers by requiring reimplementaion based on UI evaluations
- Results from lightweight evaluation may be less reliable then more extensive, controlled UI testing techniques.

7.3.5 Shared understanding vs distinct roles

Agile methods lean towards a generalist approach to software development where all of the developers not only have a shared understanding of the design but are equally qualified to work on any part of the system [2]. Benefits of this approach include improved communication between team members and increased flexibility in terms of what works is done by whom. This is most apparent in XP which is one of the more popular and development-centric approaches. Specific practices such as collective code ownership, and coding standards and pair programming encourage all developers to be collectively responsible for the system and should be equally qualified to work on any part of it [11, 12]. Usability engineering is a distinct discipline that software developers are typically not trained to do. Usability engineers are needed for projects where usability is a key quality attribute and the user interface design is non-trivial [97]. However, usability engineers are often not trained as skilled software developers.

There are several ways the conflict between the concept of generalization versus the apparent need for specialization with the integration of usability has been addressed. Patton and Mezsaros have noted that agile developers can be trained in some usability practices so they can effectively take on the roles of both developers and usability engineer [83, 102]. This sort of approach is more suited to systems usability is less important to the customer and where the user interface is relatively straightforward. One potential downside of this approach is that developers trained in this way may not be qualified to take on usability designs for more complex user interfaces. In addition, certain user-focused activities such as on-site inspections, user interviews and usability evaluations can be time-consuming activities that may limit the amount of time developers can devote to software development. In addition, this highlights the potential conflict of interest that can arise between UI design and software development that can arise. Given a limited amount of time to complete a feature, the developer is more likely to sacrifice usability to get the code implemented since functioning code is central to agile methods [13].

XSBD takes a more specialist approach where usability engineering role and software development roles are filled by different individuals from their respective disciplines (See Section 5.2). One benefit of having a separate usability engineer on the team is that she is able to bring to bear a number of user-centered activities that are important to ensure that the system meets the end user needs and customer requirements. In addition, by working in parallel with agile developers, she is able to focus on the UI design aspects of the project while developers can focus on the software design aspects. Not having to worry about the UI design on top of his other duties was viewed as a key benefit by team members on the Touch Screen team at Meridium (See Section 6.4.2: Issue F). One downside of this approach is

that having specialist roles on the team like a usability engineer makes it less flexible. Usability engineers and developers working in parallel need to carefully coordinate handoffs of designs and implementations so that each group can work effectively. In practice, close communication and use of specific practices can enable the team to maintain some flexibility in how the team functions. For example, developers can work on non-UI related feature work if they are waiting on feedback from the usability engineer. Similarly, the usability engineer can use things such as ‘click-through’ prototypes and Wizard-of-Oz testing if functionality is not handed off in a timely fashion (See Section 6.4.2: Issue F). In addition, having separate roles on the team can result in misunderstandings or conflicts between developers and usability engineers (See Section 6.3.2: Issue 1, Issue 2). These issues can be addressed by ensuring that all team members have a good high-level understanding of what each person is doing and why. For example, in the case studies at Meridium, team members appeared to be more accepting of the usability engineer after gaining an understanding of what she was doing at a high-level (See Section 6.3.2: Issue 1, Section 6.4.2: Issue B). In this way team members developed a broad but shallow understanding of the different roles on the team and a deep understanding of their respective discipline. In principle, this is similar to the concept of generalizing-specialists. However, generalizing-specialists in this case are not viewed as a temporary bridge to achieving the ultimate goal of having all team members be generalists as suggested by Ambler [2].

The tradeoffs of this approach are summarized below:

Having a distinct usability engineering role on an XSBD team

- + Can help ensure system meets end user needs by bringing someone with knowledge and experience in designing/evaluating UIs to the team
- + Can offload UI work from developers so they can focus on the code
- Makes the team less flexible in terms of how tasks can be assigned
- Can introduce communication/collaboration problems within team due to usability engineer’s lack of knowledge and experience with software development and vice versa.

7.3.6 Simplicity in the code vs simplicity in the UI

This conflict arises because of the different contexts in which usability engineers and agile software developers work. Software developers’ primary concern is with the development of functional software that meets requirements. Usability engineers’ primary concern is to ensure that the system is usable and meets end user needs. This can result in communication difficulties between the two groups due to a lack of common ground [86]. For example, for a software developer, ‘design’ likely

refers to the software code—how algorithms are implemented and data structures designed. For a usability engineer, ‘design’ likely refers to the user interface—the information and interaction design of the UI. This causes the conflict that can arise between the two groups as they try to optimize their respective areas, i.e. achieve ‘simplicity in the design’ [13, 35].

One of the key principles of agility as laid out in the Agile Manifesto is: *Simplicity--the art of maximizing the amount of work not done--is essential* [13]. Agile developers strive to develop the simplest solutions that meet the requirements. Overbuilding solutions—for example to design something to meet some hypothetical future need—can lead to wasted develop effort as requirements change [12]. Similarly, in usability engineering, simplicity in the UI design is an important principle. Straightforward designs that match users’ expectations tend to be easier to learn and use [35]. The conflict between usability engineers and agile developers comes about because simplicity in the code often does not correspond to simplicity in the UI.

Communication between team members, first and foremost, is critical to resolve misunderstandings between agile developers and usability engineers as to what is meant by terminology such as ‘simplicity in the design’. But beyond developing this common ground, they must balance their competing concerns against the customer requirements. Ultimately, the goal of the team is to develop a high quality system that meets the needs of the customer [36]. Usability is one of the quality attributes of the system. In XSBD, prioritized high-level goals are defined that relate not only to usability, but also to development concerns such as performance and reliability (See Section 5.3.2). These goals are jointly developed and agreed upon by the entire team and are revisited periodically as the system is developed incrementally. By doing so, the team can make informed decisions when the question of whether to optimize the code or the UI arise. For example, if usability is one of the high priority goals as it was in the Touch Screen project, then the team may opt to simplify and streamline the UI at the cost of additional development effort. However, if this approach is followed, team cohesion can be negatively affected since individual team members may feel that their efforts are being marginalized or ignored. This was the case in the Web Components project where usability concerns proved to be secondary to development-related concerns. The usability engineer at times had little influence within the team and had trouble working with them effectively (See Section 6.3.2: Issue 3). Addressing this problem likely depends on having committed team members who respect one another and have experience working with each other [12, 13].

The tradeoffs of this approach are summarized below:

Defining and maintaining prioritized design goals to guide design decisions through an XSBD project

- + Can help address conflicts between usability and development design concerns by unifying team efforts around core project concerns
- + Clearly defines quality attributes that are of central importance to the customer.
- Can lead to power imbalances between software developers and usability engineers

7.4 Values and principles of a combined agile usability approach

Sections 7.1, 7.2 and 7.3 describe the results of an analytic analysis of the XSBD approach. They verified that XSBD adhered to core values of both agile development methods and usability engineer methods and described how specific conflicts between agile and usability methods were addressed in XSBD. Based on the above analysis, I present a set of core principles that any agile usability approach should strive toward. These principles are derived from the principles from the Agile Manifesto, which present a good starting point and reflect the agile-centric nature of the XSBD approach in practice. Note that the values from the agile manifesto do not need to be modified as the XSBD approach is in general agreement alignment with them as shown in section 7.1.

7.4.1 New principles

The following are new agile usability principles not based on existing principles from the Agile Manifesto.

- ***Perform early and continuous usability evaluations that are linked to high-level goals to ensure that designs meet end user needs.***

Getting usability feedback early and often through user testing and ensuring that those tests were linked to high-level goals helped ensure that the UI met end user goals and the overall vision of the system. In XSBD, this was done through the development and use of the CDR (See Section 6.4.2: Issue B, Issue G).

- ***Balance the often competing needs of development and usability personnel based on project goals.***

Given their different domain areas and focus points, developers and usability engineers at times had competing or conflicting concerns. Defining, prioritizing and maintaining project goals supported in conjunction with close collaboration and communication between team members helped address this issue (See Section 6.3.2: Issue 2, Section 6.4.2: Issue A).

- ***Deliver UI designs frequently, to get continual feedback from end users and customers to improve the product.***

Continuous feedback from users is needed to ensure that the UI is meeting end user needs and helps to identify missing requirements [66, 84, 97, 127]. In XSBD, the finalized UI designs for features were delivered at the end of each iteration. However, within each iteration it was also important to share working designs and mockups to the customer and other team members for early feedback and address potential misunderstandings (See Section 6.4.2: Issue G).

- ***Respect other team members and work diligently to earn theirs.***

Beck realized the importance of respect in agile teams and added it as an XP value in the second edition of his book [12]. Similarly for agile usability teams to work effectively, it is important that no one person's work is unfairly ignored or marginalized. Openness to different practices and mindsets and working honestly with other team members is essential. Developing this mindset was important in the XSBD case studies as other team members were unaccustomed to working with usability engineers (See Section 6.3.2: Issue 3, Section 6.4.2: Issue F).

7.4.2 Modified principles

The following principles have been modified from their original formulation to better fit the underlying motivations of agile usability teams. For each I include the modified principle, the original principle and the rationale behind the change.

- ***Business people, developers and usability engineers must work together daily throughout the project.***

Original Principle: *Business people and developers must work together daily throughout the project.*

This change was made to highlight the importance of direct and continuous communication between usability engineers, developers and business people (See Section 6.4.2: Issue F). Even though usability engineers and developers are working in parallel and handing off their work to each other at the end of each iteration, daily communication is still vital to communicate changes and clear up misunderstandings [84, 127].

- ***Working software that meets high-level goals (both functional and usability) is the primary measure of progress***

Original Principle: Working software is the primary measure of progress.

This change emphasizes that in agile usability approaches like XSBD, the emphasis is not only on developing functionally correct systems, but systems that meet end user needs (See Section 6.3.2: Issue 2, Section 6.4.2: Issue A).

- ***Agile processes promote sustainable development. The sponsors, developers, usability engineers and users should be able to maintain a constant pace indefinitely.***

Original Principle: Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

This change highlights the fact that XSBD includes a distinct usability engineering role within the team that should be fully integrated and included in planning and review activities. This also addresses a common problem in organizations that use an agile usability approach: usability engineers are often overloaded and working on multiple projects simultaneously [97].

- ***Continuous attention to technical excellence and good design, both in terms of the code and usability, enhances agility.***

Original principle: Continuous attention to technical excellence and good design enhances agility.

This change was made to emphasize that this principle is important both from a developer perspective and from a usability perspective. It also highlights the fact that ‘design’ can refer to something different depending on which perspective you’re looking from [13, 35].

- ***Work towards simplicity—the art of maximizing the amount of work not done—by making appropriate tradeoffs between development and usability concerns.***

Original principle: Simplicity—the art of maximizing the amount of work not done—is essential.

Similar to the previous principle, this change was made because ‘simple’ can refer to and mean different things for a developer versus a usability engineer. Working towards simplicity will require that tradeoffs be made between the two areas [13, 35].

7.4.3 Unchanged principles

The following principles are left unchanged from the Agile Manifesto [13]:

- ***Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.***

This principle is still of central importance since the customer is the one that is paying for the system being developed. Usability is a subset of the qualities that contribute to its value to the customer [101, 135].

- ***Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.***

A core part of this research effort was to ensure that XSBD was flexible and allowed teams to still be responsive to change (See Section 6.1.1). This is a key concept of agility.

- ***Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.***

Agile methods rely on incremental development cycle so they can get feedback from customers as early and as often as possible. XSBD was developed to work within this framework (See Section 5.4).

- ***Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.***

Ultimately, the people within the agile team and how well they work together will be the best predictor of whether a project will be successful [12]. This is true of any team that uses XSBD. XSBD represents one of the tools and helps develop the environment through which a team can succeed.

- ***The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.***

This proved to be the case in both XSBD case studies at Meridium (See Section 6.3.2: Issue 5, Section 6.4.2: Issue D). Many of the optimizations to the CDR and how it is used depend on it.

- ***The best architectures, requirements, and designs emerge from self-organizing teams.***

In using the action research case study approach, I was able to improve the XSBD process and how it was implemented by actively engaging the team members (See Section 6.1.1). They often made their own changes to how they operated based on problems that came up.

- *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*

The idea of continuous feedback to improve the product and identify new and changing requirements is a core concept of both agile and usability methods (and by extension XSBD) [12, 111]. This idea has been extended to the development team itself—continuously improving how they are working together. This is achieved through retrospectives and by using the action research case study methodology in this research effort.

7.5 Chapter summary

This chapter described the results of an analytic evaluation of the XSBD approach whose purpose was to show how it adhered to key agile and usability values. I first described how XSBD is an agile approach by making clear how it aligned with key values of agility—drawing on specific practices of XSBD and examples from the action research case studies at Meridium. I then detail how XSBD is a valid usability engineering approach by showing how it follows key principles and practices common to usability in general. Third, I describe how XSBD addressed apparent divergence points between agile and usability methodologies through specific practices and highlight the tradeoffs of those practices with respect to core agile and usability values. Based on the above analysis and the results of the action research case studies, I present a series of general agile usability principles based on the principles from the Agile Manifesto.

8 Conclusions and future work

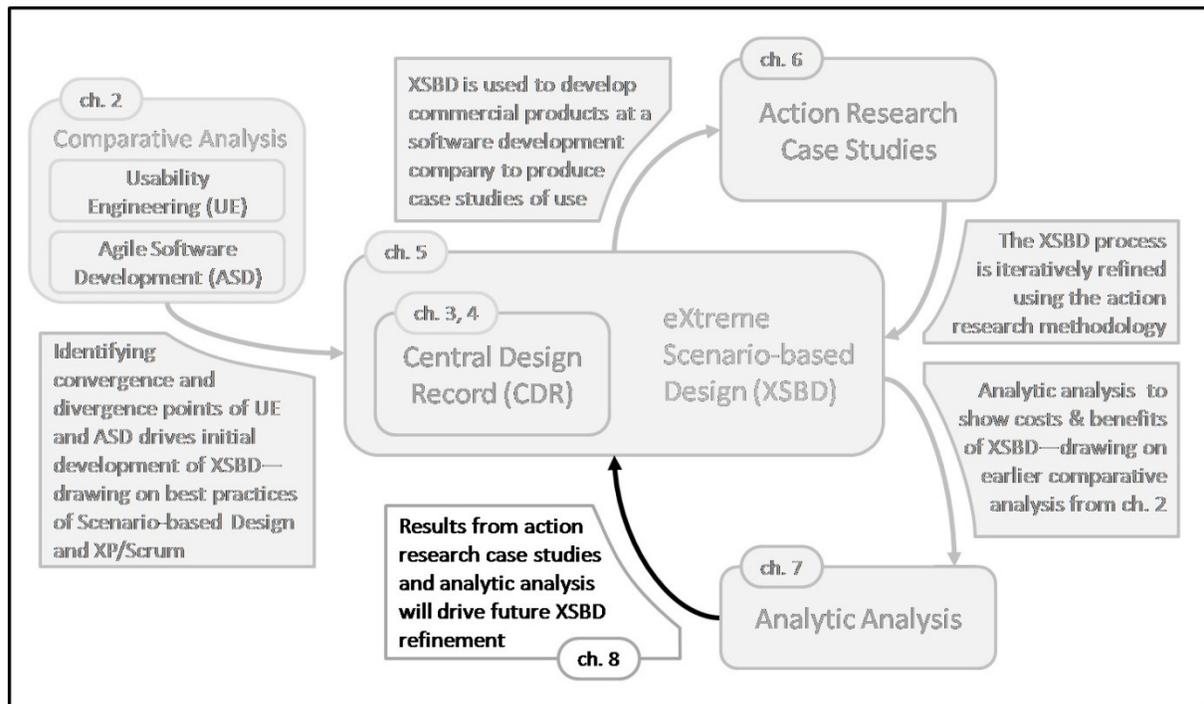


Figure 40. Overview of solution approach. Chapter 8 will summarize key findings of this research effort and directions for future work.

As agile organizations have begun to develop more user-facing, UI-intensive systems, they have identified the need to find ways to develop more usable systems without sacrificing key benefits of agile methods usable [4, 56, 102]. My development of the XSBD approach is motivated by the need to find ways to integrate usability into agile software development teams to satisfy this need.

My XSBD approach provides key usability benefits of the scenario-based design approach—an established usability engineering process developed by Carroll and Rosson—and is compatible with an agile development framework modeled on leading agile processes like XP and Scrum [12, 115, 116]. The XSBD approach was designed for use in projects in which a large part of the overall quality of the system will be determined by how usable it is. Software development and usability design proceeds in parallel, allowing the dedicated usability and development personnel to work together quickly and efficiently. This requires close communication and careful coordination of the disparate usability and agile development work practices. A core aspect of XSBD is the central design record (CDR), which is the shared design representation that guides usability development. It tightly couples usability evaluation results to design features and high-level project goals, allowing the usability engineer to

leverage key practices (and the resulting benefits) of SBD while working in an agile incremental development cycle.

I began developing the XSBD approach at Virginia Tech—evaluating and improving it through several student-led development efforts. To improve and demonstrate the applicability of XSBD, I partnered with Meridium, Inc., a software and services company based in Roanoke, Virginia. Using an action research case study approach, I worked with several development teams there who used the XSBD approach to develop commercial products. These development efforts helped me to further refine the XSBD approach, balance key tradeoffs between usability and agile development concerns and demonstrate its utility in practice. In addition, I demonstrated that XSBD is a valid agile usability approach by showing how it aligns with core values of both agility and usability. In this analysis, I discuss how divergence points between agility and usability were addressed through specific XSBD practices and discuss their tradeoffs with respect to balancing usability and agile concerns. Based on this analysis and the action research case studies, I provide a set of agile usability principles to provide further guidance to agile and usability practitioners in general.

In this chapter I will first summarize key findings of my work in XSBD (See Figure 40). I will then discuss the contributions of this work with respect to both industry and academia. I will close with a discussion of future work to further improve XSBD and expand its focus to include a broader range of research and software development concerns.

8.1 Key findings

This section summarizes the key findings of this research effort stated in terms of the four key challenges facing an integrated agile usability approach that were introduced in Chapter 1 (See Section 1.1).

1. **Comparison of core principles.** In considering usability methods and agile methods, what are the convergence and divergence points from a high-level principles perspective? It is important to identify where difficulties lie in integrating the two approaches, where similarities can be exploited in a combined approach and what the costs and benefits of the resulting approach are.
 - XSBD leverages key similarities between agile and usability methodologies to integrate scenario-based design into an agile framework (See Section 2.3.1). For example both approaches use cyclical development processes that involve continuous design-feedback loops. In XSBD, by using the CDR and compressing the phased design cycles of SBD, usability work can be done within an agile incremental development cycle (See Section 7.2).

In addition, both agile and usability methods rely on regular testing to validate designs and inform future development effort. XSBD relies on lightweight usability evaluations to continuously evaluate the system as it is incrementally developed (See Section 5.3.4). Finally, both agile and usability are human-centered in that they rely on regular communication and interaction with relevant stakeholders. XSBD relies on close communication between key stakeholders including usability engineers, developers, customers and end users (See Section 5.2). Not only do these similarities make it easier to integrate usability practices into an agile framework, they make it easier for agile practitioners to understand and accept usability practices.

- Conflicts between agile and usability methods do not represent insurmountable philosophical differences between the two areas. Rather, they represent differences in context, mindset and practices. In XSBD, these conflicts—such as the agile focus on customer satisfaction versus the usability focus on the end user—are addressed through specific methods that draw on a combination of both agile and SBD techniques highlighted in Chapter 6. Tradeoffs of those approaches are discussed in Chapter 7 with the understanding that the ultimate goal of any agile usability project is to deliver a high quality system that satisfies the needs of the customers (See Section 7.3). Usability is one of the factors contributing to system quality and its importance to the customer will determine whether XSBD is appropriate for use in any particular project.
2. **Balance of power.** In developing a specific instantiation of a combined approach, how should the approach be balanced between usability and agile methods? For instance, an agile-centric approach may involve providing some level of usability-training to agile developers while a usability-centric approach may have usability engineers complete UI designs before agile developers begin their work. The integration approach will have an impact on the long-term viability and acceptance of an integrated process in practice.
- In many commercial organizations like Meridium, it is more effective to take an agile-centric integration approach. Agile methodologies are being widely adopted in practice and take a customer-centric approach—focusing development practices around delivering value to customers. The most effective way of integrating usability practices into such an organization is through an agile-centric approach that leverages similarities between agile and usability approaches (See Section 6.3, 6.4). It needs to ensure that changes to established practices at the agile organization are centered on delivering value to the customer. This is done in XSBD

through the CDR which helps define how important usability is with respect to other software quality attributes (See Section 5.3.2).

- The overall usability of the system can be improved by following an accelerated, agile-centric development approach. By using lightweight evaluation techniques, the usability engineer was able to run usability tests within a single iteration and provide actionable guidance to developers in a timely manner (See Section 6.4.2: Issue G). This way, the usability engineer was able to get feedback from users earlier and more often in the development process. By using the CDR, the usability engineer focused those evaluations on critical design features through claims linked to high-level design goals which ensured that UI changes met the needs of end users. In addition, since agile methods work by lowering the cost of change curve, these improvements can be made efficiently throughout the development process [69].
 - Even though XSBD takes an agile-centric approach in that usability practices are molded to fit within an incremental development cycle, the actual influence of usability with respect to other quality attributes will differ depending on the customer and the system being built. For example, in the Web Components project, the usability engineer had less influence as the team focused more on functionality and performance (See Section 6.3.3). In the Touch Screen project, usability was a critical quality attribute and thus had a larger impact on what features were developed and when (See Section 6.4.2: Issue A). Effectively using XSBD requires a period of adjustments and refinements. Using an action research methodology was effective in this research effort in that it iteratively improved the XSBD process to fit the needs of each team while simultaneously contributing to the research needs of this project.
3. **Checks and balances.** How can appropriate tradeoffs between the agile and usability practices in the integrated approach be made? Any integrated agile usability approach will have tradeoffs that may, for example, sacrifice development velocity for validation of the usability of a user interface. The integrated approach should help agile usability teams make these tradeoffs given the specific needs of their development project.
- Prioritized goals defined at the start of the project, agreed upon by all team members and followed through to project completion are important to ensure balanced design decisions that take into account customer, development, and usability needs (See Section 6.5). These goals, which represent a core part of the CDR, need to be revisited on a regular basis through a unified planning and assessment process as the product is incrementally developed to ensure that goals are being met and that adjustments to the goals themselves are made as needed (See

Section 6.4.2: Issue A). Claims can be used to capture alternate design proposals when team members disagree about how features can be designed (Section 6.4.2: Issue F). They can later be tested by the usability engineer to determine which design is optimal.

- In XSBD, there are specialized roles (e.g. usability engineer, developer, QA tester, etc) because different areas of focus--in this case software development and usability--require different knowledge bases and skill sets (See Section 5.2). It is important that all XSBD team members have a broad but shallow understanding of what the rest of the team members were doing in addition to having a deep understanding of their own particular area (See Section 6.4.2: Issue B). This shared understanding, achieved through regular communication and sharing of information in the CDR, allowed team members to work together more effectively by understanding the needs of others and where their work lies with respect to the project as a whole.
 - Having respect for other team members is critical to ensure that the XSBD team works together effectively and properly balances development and usability concerns. Because usability and development work in parallel and have different work practices and motivations, developers and usability engineers can come into conflict with one another. This was especially the case in both development efforts in Meridium where other team members were unaccustomed to working with a usability engineer and did not understand usability work practices and motivations (See Section 7.4.1). Continuous communication and collaboration between software and usability engineers combined with early and continuous demonstration of their value to the team effort helped to mitigate this issue.
4. **Synchronicity.** How can the agile development and usability engineering sides of a project stay synchronized throughout the development effort? This is critical as design drift between the interface design and its implementation can render the usability work irrelevant and result in a system that has poor usability. This problem is exacerbated by the speed at which agile development projects proceed. Addressing this issue depends on proper development and use of shared design representations, communication practices and process planning structures within the team.
- The XSBD approach synchronizes development and usability through regular checkpoints at the start and end of iterations. At the end of each iteration, specific handoffs between the developers and usability engineer need to occur (See Section 6.4.2: Issue E). Typically, this will involve the usability engineer handing off the design for the developers to implement in

the next iteration and the developers handing off the implemented design for the usability engineers to evaluate in the next iteration. This requires a unified planning process where both development and usability tasks are managed together.

- In addition to regular synchronization points at the start and end of iterations, agile developers and usability engineers need to communicate with one another within each iteration as needed. Developers will need input from usability engineers to clear up misunderstandings about and fix holes in the delivered design (See Section 6.4.2: Issue F). Usability engineers will need to work with developers to identify discontinuities between implementations and the original UI designs. These types of interactions are critical as they enable the XSBD process to remain flexible and address changing requirements and technical limitations which could otherwise lead to design drift—even within a single iteration.
- Share only the most important and easily understandable design representations as needed with other team members. Representations used in the CDR such as scenarios and claims are no substitute for regular communication between team members. In XSBD, we found that high-level design goals, scenarios and design mockups were the most useful and easily understandable usability design representations to share with other team members (See Section 6.3.2: Issue 5, Section 6.4.2: Issue D). Developers and QA personnel relied on scenarios and mockups in particular in developing and testing the functioning system. Usability results and claims could be summarized and communicated by the usability engineer most easily through face-to-face communications.

8.2 Contributions of this research effort

The XSBD process was developed to address a pressing need within the agile community—finding ways to develop more usable products that are compatible with existing agile values. It also advances the body of knowledge within the software engineering and usability research communities by bridging academic research and leading industry practices. Key contributions of this work are summarized below both from an industry perspective and a research perspective.

8.2.1 Contributions to the agile community

This section details my contributions to agile practitioners and the software development community in general through this effort.

Detailed description of agile usability approach—XSBD. XSBD is a specific instantiation of agile usability that has been substantiated in practice (See Chapter 5). It provides actionable guidance to practitioners who want to integrate usability into their agile organizations. XSBD provides a framework in which agile developers, usability engineers, customer representatives and end user representatives can communicate with one another and coordinate their work to efficiently develop a high quality system.

Central Design Record as shared design representation. Usability engineers can develop and use the CDR to help them use a scenario-based usability design process within an agile framework (See Section 5.3). It helps them develop the UI incrementally, guide usability evaluations, ensure that the system meets high-level project goals and share relevant UI artifacts with other team members.

Detailed case studies of XSBD use in practice. Several detailed case studies of XSBD can provide utility not only to those interested in adopting it in practice, but also to agile usability practitioners in general (See Chapter 6). These case studies describe the integration of XSBD within an agile organization with limited previous exposure to usability practices—a situation that is relatively common within the agile community. In addition, it was used by both a collocated team and a distributed team. Agile methods are increasingly being used in distributed teams and information on how usability can be integrated in such a context could prove useful. Finally, these case studies, in combination with other case study work in agile usability, contributes to the growing corpus of knowledge from which the agile community can draw on.

Agile usability principles. Generalized guidance to agile practitioners is provided in the form of a set of agile usability principles (See Chapter 7). This set of principles is meant as an addendum to the Agile Manifesto for those that are implementing an agile usability approach. These principles are based on results drawn from the use of XSBD in practice and lessons learned from other agile usability practitioners [3, 64, 66, 46, 84, 127]. They are meant to provide higher level guidance for those that are adapting XSBD for their specific organization or using a different agile usability approach.

Introduction of agile usability to new domain. The XSBD approach has been used in a number of development efforts at Meridium, Inc., an agile software development organization with limited previous experience with usability processes. This effort represents the start of the introduction of agile usability within their domain area of asset performance management. Meridium is now

committed to making usability a key part of its development practices using XSBD and is using it as a way to maintain competitive advantage over competitors.

8.2.2 Contributions to the research community

This section summarizes my contributions to the HCI, usability and software development research communities.

Addressing theory-practice gap. This work addresses the theory-practice gap by bringing together usability research and agile practitioner communities using an action research methodology (See Section 2.1.2) [7, 23, 26, 101, 135]. I developed XSBD at Virginia Tech based on established research in scenario-based design, design rationale and design knowledge reuse [24, 26, 91, 92, 99]. I instantiated XSBD in practice through a partnership at Meridium and provided them with actionable guidance while iteratively refining XSBD and evaluating it from a theoretical perspective. Results from the case studies in addition to the continued partnership with Meridium can serve as a base for further collaborations in those research areas.

Contributing to general body of software and usability knowledge. This work also contributes to the general body of knowledge with software engineering and usability engineering focusing specifically on the challenges of integrating usability into an agile software development framework. It has been published at a number of leading agile and HCI conferences and journals [61, 64, 65, 66, 67, 78]. This complements similar research being done to integrate usability and software engineering in general [10, 103, 109, 122].

Design rationale in practice. This work contributes to the general research area of design rationale use [20, 25, 27, 47, 70, 126, 130, 131]. It provides a specific example of how design rationale can be used in practice to design usable systems. Specifically, it highlights how claims can be used in a lightweight manner to guide development decisions and focus usability evaluations. This relates back to my past work in design knowledge reuse and provides a gateway to further research in this area [62, 63, 67, 79, 89].

8.3 Future work

The development and use of the XSBD process in practice represents an initial contribution to how usability can be integrated into an agile organization which in itself is a complex and multifaceted

problem. The action research case studies and analytic evaluation of the approach have identified areas for further study which are summarized below.

Additional case studies of agile usability in practice. Additional case studies of agile usability approaches like XSBD are needed to further evaluate what types of systems and organizations they are best suited towards. Although the case studies at Meridium have shown that the XSBD approach can be used in practice to efficiently develop usable systems, different software organizations each have their own unique domains, development cultures and set of practices that may need to adapt XSBD in different ways to use it effectively. For example, one open question is how well the XSBD approach scales to larger development efforts. This problem is not specific to XSBD but to agile approaches in general [17]. In addition, future efforts could study how the approach could be used to develop different types of things such as life-critical systems.

Integrating with other stakeholders. This research focused on integrating development and usability concerns. However, there are a number of specific stakeholders whose work is interdependent with that of agile developers and usability engineers. For example, many organizations have dedicated product managers, quality assurance personnel, security specialists, and documentation personnel who need to interact with developers and usability engineers on a regular basis as they work together to develop the system [See Section 6.3.3, 6.4.3]. Further work is needed to integrate these additional personnel and their specific concerns and practices into an XSBD team. For example, a more in-depth understanding of the similarities and conflicts between usability testing and quality assurance testing is needed to ensure that they can be made to effectively function synchronously.

Social issues in agile usability. The people in an agile usability team and how well they communicate with one another is a good indicator of whether the product will be successfully developed. In this effort, I found that interpersonal relationships between team members—particularly conflicts that could occur between software developers and usability engineers—had a significant effect on how well the team functioned. Future work could leverage research from the social sciences to study factors that determine whether or not an agile usability team will work together effectively and help identify what can be done to mitigate problems early on. Whitworth and Biddle have done some work in this area [134]. Barksdale et al have begun to explore social issues specifically within agile usability teams—building on some of my work in XSBD [8].

Tool support for XSBD. Future work could include developing tools to support the use of XSBD as it is increasingly used in practice. For example, a tool could be developed to manage the CDR and integrate it with existing project management tools like Microsoft Visual Studio® or agile-specific management tools. This could help in scaling XSBD to larger projects. In addition, tool support could provide a link back to my previous work in design knowledge reuse tools [62, 63, 67, 79, 89]. This would require revisiting questions such as what information should be captured, how the information should be stored, how it can be maintained [6, 20, 112]. Specifically, Wahid et al are continuing to do work on claims-centric knowledge reuse tools that could serve as a base for a future CDR management tool [130, 131, 132].

8.4 Summary

XSBD is a specific instantiation of an agile usability approach that integrates scenario-based design into an agile development framework. It addresses the need within the agile community to find ways to integrate usability into agile teams. XSBD allows usability engineers to leverage key benefits of SBD while working within an agile incremental development framework by using the CDR—a shared design representation that directly links high-level goals to design features and usability evaluation results.

In this work, I have identified key *convergence* and *divergence* points between agility and usability to guide the development of XSBD and highlight tradeoffs of the approach with respect to key values and principles of both areas. By leveraging key similarities in the two approaches and to facilitate adoption within the agile community, I developed an *agile-centric* integration approach in which usability practices are designed to work within an agile framework. Using an action research methodology, I partnered with Meridium to use XSBD in practice to develop real products for customers. This way, I was able to focus on identifying how XSBD can be used to *properly balance* differing usability and development concerns and how usability engineers and agile developers can work *synchronously* to efficiently develop a usable end product.

XSBD provides actionable guidance to agile practitioners who wish to adopt an agile usability development approach. Meridium, which previously had limited exposure to usability practices, is fully committed to adopting this agile usability approach which is an encouraging beginning for the approach in practice. The case studies themselves will be of value to both agile practitioners and researchers in usability and software engineering. I have published results from this work in leading agile and HCI conferences [61, 64, 66]. In addition, I have developed an initial XSBD tutorial to

introduce the approach to usability and agile practitioners [65]. I will continue to work to improve XSBD and encourage its adoption in practice.

9 References

1. Ambler, S. W. (2008). Agile practices and Principles Survey Results: July 2008. Ambyoft, Inc. Available: <http://www.ambysoft.com/surveys/practicesPrinciples2008.html>
2. Ambler, S. W. (2003). Generalizing specialists: Improving your IT career skills. Ambysoft, Inc. <http://www.agilemodeling.com/essays/generalizingSpecialists.htm>
3. Ambler, S. W. (2007). Introduction to Agile Usability: User Experience Activities on Agile Development Projects. Ambysoft, Inc. Available: <http://www.agilemodeling.com/essays/agileUsability.htm>
4. Ambler, S. W. (2008). Tailoring Usability into Agile Software Development Projects. In Law, E., Hvannberg, E. and Cockton, G. (Eds.). *Maturing Usability: Quality in Software, Interaction and Value*. Springer-Verlag, London, England.
5. Ambler, S. W., and Jeffries, R. (2002). *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. John Wiley & Sons, Inc., New York, NY.
6. Arnold, R. S., Slovin, M., and Wilde, N. (1993) Do design records really benefit software maintenance. In *proc. Software Maintenance '93*, 234-243.
7. Avison, D. E., Lau, F., Myers, M. D., and Nielsen, P. A. (1999). Action Research: Making Academic Research Relevant. *Communications of the ACM*, 42 (1), 94-97.
8. Barksdale, J. T., Ragan, E. D., and McCrickard, D. S. (2009). Easing Team Politics in Agile Usability: A Concept Mapping Approach. In *Proc. Agile '09*, 19-25.
9. Baskerville, R. L. (1999). Investigating information systems with action research. *Communications of the AIS* 2(3es).
10. Bass, L., and John, B. E. (2003). Linking usability to software architecture patterns through general scenarios. *Journal of Systems and Software*. 66(3), 187-197.
11. Beck, K. (1999). Embracing change with extreme programming, *Computer*, 32(10), 70-77.
12. Beck, K. (2004). *Extreme Programming Explained: embrace change* (2nd Edition). Addison-Wesley Longman Publishing Co., Inc., Boston, MA.
13. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001). *The Agile Manifesto*. Available: <http://agilemanifesto.org>
14. Beyer, H., Holtzblatt, K., and Baker, L. (2004). An Agile Customer-Centered Method: Rapid Contextual Design, *XP/Agile Universe '04*, 50-59.
15. Bhatia, S., Dahn, C., Lee, J. C., Sampat, M., and McCrickard, D. S. (2006). VTAssist--A location-based feedback notification system for the disabled. In *proc. ACMSE '06*, 512-517.
16. Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. *Computer*, 21(5), 61-62.
17. Boehm, B. W. (2002). Get ready for agile methods with care. *Computer*, 35(1), 64-69.
18. Borchers, J. O. (2000). A Pattern Approach to Interaction Design. In *Proc. DIS 2000*, 369-378.
19. Brooks Jr., F. P. (1995). *The Mythical Man-Month: Essays on Software Engineering* (20th Anniversary Edition), Addison-Wesley Professional, Boston, MA.
20. Buckingham Shum, S., and Hammond, N. (1994). Argumentation-based design rationale: what use at what cost? *IJHCS* 40(4).
21. Card, S. K., Newell, A., and Moran, T. P. (1983). *The psychology of human-computer interaction*. L. Erlbaum Associates, Mahwah, NJ.

22. Carroll, J. M. (Ed.) (2003). *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*. Morgan Kaufmann Publishers, San Francisco, CA.
23. Carroll, J. M. (1991) *The Kittle House Manifesto*. In J. M. Carroll (Ed.), *Designing Interaction: Psychology at the human-computer interface*. Cambridge University Press, New York, NY.
24. Carroll, J. M. (2000). *Making Use: Scenario-Based Design of Human-Computer Interactions*. The MIT Press, Cambridge, MA.
25. Carroll, J. M., and Campbell, R. L. (1986). Softening Up Hard Science: Reply to Newell and Card. *Human-Computer Interaction* 2, 227-249.
26. Carroll, J. M., and Kellogg, W. A. (1989). Artifact as theory-nexus: Hermeneutics meets theory-based design. In *proc. CHI '89*, 7-14.
27. Carroll, J. M. and Rosson, M. B. (2003). Design rationale as theory. In Carroll, J. M. (Ed.) *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*. Morgan Kaufmann Publishers, San Francisco, CA.
28. Carroll, J. M. and Rosson, M. B. (1992). Getting around the task-artifact cycle: how to make claims and design by scenario. *ACM Transactions on Information Systems (TOIS)*, 10(2), 181-212.
29. Carroll, J. M. and Rosson, M. B. (1987). *Paradox of the active user. Interfacing though: cognitive aspects of human-computer interaction*, MIT Press, Cambridge, MA, 1987.
30. Chamberlain, S., Sharp, H., and Maiden, N. (2006). Towards a Framework for Integrating Agile Development and User-Centred Design., in *Proc. XP '06*, 143-153.
31. Chewar, C. M., Bachetti, E., McCrickard, D. S., and Booker, J. (2004). Automating a design reuse facility with critical parameters: Lessons learned in developing the LINK-UP system. In *Proc. CADUI '04*, 236-247.
32. Chewar, C. M. and McCrickard, D. S. (2003). Educating novice developers of notification systems: Targeting user-goals with a conceptual framework. In *Proc. ED-MEDIA '03*, 2759-2766.
33. Chewar, C. M., McCrickard, D. S., and Sutcliffe, A. G. (2004). Unpacking critical parameters for interface design: Evaluating notification systems with the IRC framework. In *Proc. DIS '04*, 279-288.
34. Constantine, L. L. (2001). *Process Agility and Software Usability: Toward Lightweight Usage-Centered Design*. *Information Age*, 8(2). Reprinted in L. Constantine (Ed.), *Beyond Chaos: The Expert Edge in Managing Software Development*. Addison-Wesley, Boston, MA.
35. Constantine, L. L., and Lockwood, L. (1999). *Software for Use: A Practical Guide to the Models and methods of Usage-Centered Design*. Addison-Wesley, Boston, MA.
36. Cockburn, A. (2007). *Agile Software Development: The Cooperative Game (2nd Edition)*. Pearson Education.
37. Cooper, A. (2004). *The Inmates are Running the Asylum (2nd Edition)*. Pearson Higher Education.
38. Cooper, A., and Reimann, R. (2003). *About Face 2.0: The Essentials of Interaction Design*, Wiley Publishing Inc., Indianapolis, IN.
39. Davison, R. M., Martinsons, M. G., and Kock, N. (2004). Principles of canonical action research. *Information Systems Journal*, 14(1).
40. Dingsøyr, T., Hanssen, G. K., Dybå, T., Anker, G., and Nygaard, J. O. (2006). Developing Software with Scrum in a Small Cross-Organizational Project. In *Proc. EuroSPI '06*: 5-15.
41. Engelberg, D. and Seffah, A. A Framework for Rapid Mid-Fidelity Prototyping of Web Sites. In *Proc. In Proc. IFIP 17th World Computer Congress*, 203-215.
42. Erickson, T. (2000). *Lingua Francas for Design: Sacred Places and Pattern Languages*. In *Proc. DIS '00*, 357-368.
43. Ferreira, J., Noble, J., and Biddle, R. (2007). Agile Development Iterations and UI Design. In *Proc. Agile '07*, 50-58.

44. Flanagan, J.C. The critical incident technique. *Psychological bulletin*, 51(4), 327-358, 1954.
45. Fowler, M. Is design dead? (2002). In Auer, K., and Miller, R. *Extreme programming examined*, Addison-Wesley, Boston, MA.
46. Fox, D., Sillito, J., and Maurer, F. (2008). Agile Methods and User-Centered Design: How These Two Methodologies Are Being Successfully Integrated In Industry. In *Proc. Agile '08*, 63-72.
47. Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: elements of reusable object-oriented software*. Addison-Wesley Longman. Boston, MA.
48. Gould, J. D., and Lewis, C. (1985). Designing for usability: Key principles and what designers think. *Communications of the ACM*, 28(3), 300-311.
49. Greenberg, S., and Rounding, M. (2001). The notification collage: posting information to public and personal displays. In *Proc. CHI '01*, 514-521.
50. Gross, J. B., Daughtry III, J. M., and Lee, J. C. (2008). Heurists of the World Unite! Merging Agile Methods in Software and Interaction Design. *Agile Journal*, 3(2).
51. Harker, S. (1991). Requirements Specification and the Role of Prototyping in Current Practice. In J. Karat (Ed.), *Taking Software Design Seriously*. 339-354, Academic Press, London, UK.
52. Hartson, H. R., Castillo, J. C., Kelso, J., and Neale, W. C. (1996). Remote evaluation: the network as an extension of the usability laboratory. In *Proc. CHI '96*, 228-235.
53. Highsmith, J., and Cockburn, A. (2001). *Agile Software Development: The Business of Innovation*. *IEEE Computer*, 24(9), 120-122.
54. Hix, D., and Hartson, H. R. (1993). *Developing User Interfaces: Ensuring Usability through Product and Process*, John Wiley & Sons, Inc., New York, NY.
55. Ivory, M. Y., and Hearst, M. A. (2001). The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys*, 33(4), 470-516.
56. Jokela, T., and Abrahamsson, P. (2004). Usability Assessment of an Extreme Programming Project: Close Co-operation with the Customer Does not Equal Good Usability. In *Proc. PROFES '04*, 393-407.
57. Kelly, S., Hood, B., Lee, J. C., Sampat, M., Lally, L., and McCrickard, D. S. Enabling Opportunistic Navigation in Location-Based Notification Systems. In *Proc ACHI '09*, 32-37.
58. Kerth, N. L. (2002). *Project Retrospectives: A Handbook for Team Reviews*. Dorset House Publishing Company Inc., New York, NY.
59. Kniberg, H. (2007). *Scrum and XP from the Trenches*. InfoQ Enterprise Software Development Series. Available at: <http://www.infoq.com/minibooks/Scrum-xp-from-the-trenches>
60. Landauer, T. K. (1991). Let's get real: a position paper on the role of cognitive psychology in the design of humanly useful and usable systems. In Carroll, J. M. (Ed.), *Designing Interaction: Psychology at the human-computer interface*. Cambridge University Press, New York, NY.
61. Lee, J. C. (2006). Embracing Agile Development of Usable Software Systems. Doctoral consortium paper in *Proc. CHI '06*, 1767-1770.
62. Lee, J. C., Chewar, C. M., and McCrickard, D. S. (2005). Image is Everything: Advancing HCI Knowledge and Interface Design Using the System Image. In *Proc. ACMSE '05*, Vol. 2, 376-381
63. Lee, J. C., Lin, S., Chewar, C. M., McCrickard, D. S., Fabian, A., and Jackson, A. (2004). From Chaos to Cooperation: Teaching Analytic Evaluation with LINK-UP. In *proc. E-Learn '04*, 2755-2762.
64. Lee, J. C., and McCrickard, D. S. (2007). Towards Extreme(ly) Usable Software: Exploring Tensions Between Usability and Agile Software Development. In *Proc. Agile '07*, 59-70.
65. Lee, J. C., and McCrickard, D. S. (2009). Working in Harmony: Integrating the efforts of usability engineers and agile software developers. Tutorial presented at *ACHI '09*.

66. Lee, J. C., Stevens, T. K., and McCrickard, D. S. (2009). Examining the Foundations of Agile Usability with eXtreme Scenario-based Design. In Proc. Agile '09, 3-10.
67. Lee, J. C., Wahid, S., McCrickard, D. S., Chewar, C. M., and Congleton, B. (2007). Understanding Usability: Investigating an Integrated Design Environment and Management System. *International Journal of Information Technology and Smart Education (ITSE)*, 4(3), 161-175.
68. Leuf, B. and Cunningham W. (2001). *The Wiki Way: Quick Collaboration on the Web*, Addison-Wesley, Boston, MA.
69. Koch, A. S. (2004). *Agile Software Development: Evaluating The Methods For Your Organization*. Artech House Publishers, Boston, MA.
70. MacLean, A., Young, R., Moran, T. (1989). Design rationale: The argument behind the artifact. *Proceedings of CHI '89*, 247-252.
71. Majchrzak, L., Cooper, L. P., and Neece, O. E. (2004). Knowledge Reuse for Innovation. *Management Science*, 50(2), 174-188.
72. Martin, A., Biddle, R., and Noble, J. The XP Customer role in practice: three studies. In Proc. Agile '04, 42-54, 2004.
73. Martin, A., Noble, J., and Biddle, R. "Programmers are from Mars, customers are from Venus: a practical guide for customers on XP projects. Proc. 2006 conference on pattern languages of programs (Plop '06), ACM Press, 2006.
74. Martin, A., Biddle, R. and Noble, J. The XP Customer Team: A Grounded Theory. In Proc. Agile '09, 57-64.
75. Martin, A., Biddle, R., and Noble, J. XP Customer Practices, A Grounded Theory. In Proc. Agile '09, 33-40.
76. McCrickard, D. S., Catrambone, R., Chewar, C. M., and Stasko, J. T. (2003). Establishing Tradeoffs that Leverage Attention for Utility: Empirically Evaluating Information Display in Notification Systems. *International Journal of Human-Computer Studies*, 8(5), 547-582.
77. McCrickard, D. S., Chewar, C. M., Somervell, J.P., and Ndiwalana, A. (2003). A Model for Notification Systems Evaluation—Assessing User Goals for Multitasking Activity. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 10(4), 312-338.
78. McCrickard, D. S., Sampat, M., and Lee, J. C. (2008). Building Applications to Leverage Location Awareness: New Approaches to the Design, Implementation, and Evaluation of Mobile and Ubiquitous Interfaces. In Theng, Y. and Duh, H. (eds.). *Ubiquitous Computing: Design, Implementation and Usability*, 253-264, IGI Global.
79. McCrickard, D. S., Wahid, S., Lee, J. C., and Polys, N. F. (2005). Use and Reuse in Information and Interaction Design. In Proc. HCII '05, 8 pages (CD-ROM).
80. McInerney, P., and Maurer, F. (2005). UCD in agile projects: dream team or odd couple? *Interactions* 12(6), 19-23.
81. Medlock, M. C., Wixon, D., Terrano, M., Romero, R. L., and Fulton, B. (2002). using the RITE method to improve products: a definition and a case study. In Proc. UPA '02.
82. Meridium, Inc. (2010). About Meridium. Available: <http://www.meridium.com/company/index.asp>
83. Meszaros, G., and Aston, J. (2006). Adding Usability Testing to an Agile Project. In Proc. Agile '06, 289-294.
84. Miller, L. (2005). Case Study of Customer input For a Successful Product. In Proc. Agile '05, 225-234.
85. Miller, L. and Sy, D. Agile user experience SIG. In proc. extended abstracts of Conference on Human Factors in Computing Systems (CHI '09), ACM Press, 2009, 2751-2754.

86. Monk, A. (2003). Common Ground in Electronically Mediated Communication: Clark's Theory of Language Use. In Carroll, J. M. (ed) HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science, Morgan Kaufman, San Francisco, CA.
87. Nair, S., Kumar, A., Sampat, M., Lee, J. C., and McCrickard, D. S. (2006). Alumni Campus Tour: Capturing the Fourth Dimension in Location Based Notification Systems. In Proc. ACMSE '06, 500-505.
88. National Science Foundation. (2007). Small Business Innovation Research (SBIR) & Small Business Technology Transfer (STTR) Programs. Available: <http://www.nsf.gov/eng/iip/sbir/>
89. Ndiwalana, A., Lee, J. C., Smith, J. L., Wahid, S., Hobby, L., Chewar, C. M., and McCrickard, D. S. (2005). From Personas to Design: Creating a Collaborative Multi-disciplinary Design Environment. In Proc. HCII '05, 10 pages (CD-ROM).
90. Nelson, E. (2002). Extreme Programming vs. Interaction Design, Fawcette Technical Publications. Available: http://web.archive.org/web/20060613184919/www.fawcette.com/interviews/beck_cooper/
91. Newman, W. (1994). A preliminary analysis of the products of hci research, using pro forma abstracts. In proc. CHI '94, 278-284.
92. Newman, W. (1997). Better or Just Different? On the Benefits of Designing Interactive Systems in terms of Critical Parameters. In Proc. DIS '97, 239-245.
93. Nielsen, J. (1994). Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier. Available: http://www.useit.com/papers/guerrilla_hci.html
94. Nielsen, J. (2005). Severity Ratings for Usability Problems. Available: <http://www.useit.com/papers/heuristic/severityrating.html>.
95. Nielsen, J. (1993). Usability Engineering. Morgan Kaufman, San Diego, CA.
96. Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In Proc. CHI '94, 152-158.
97. Nodder, C. and Nielsen, J. (2008). Agile usability: best practices for user experience on agile development projects. Nielsen Norman Group, Fremont, CA, 2008.
98. Norman, D. A. (2005). Do companies fail because their technology is unusable? Interactions, 12(4), 69.
99. Norman, D. A. (1986). Cognitive engineering. In D. A. Norman & S. W. Draper (Eds.) User Centered Systems Design: New Perspectives on Human-Computer Interaction, Lawrence Erlbaum Associates, New Jersey.
100. Obendorf, H. and Finck, M. (2008). Scenario-based usability engineering techniques in agile development processes. Case study in Proc. Conference on Human Factors in Computing Systems (CHI '08), ACM Press, 2008, 2159-2166.
101. Olsen, G. (2005). The emperor has no lab coat, Interactions, 9(4), 13-17.
102. Patton, J. (2002). Hitting the target: adding interaction design to agile software development. In Proc. OOPSLA '02, 1-ff.
103. Pawar, S. A. (2004). A Common Software Development Framework For Coordinating Usability Engineering and Software Engineering Activities. Master's Thesis, Virginia Tech.
104. Payne, C., Allgood, C. F., Chewar, C. M., Holbrook, C., and McCrickard, D. S. (2003). Generalizing Interface Design Knowledge: Lessons Learned from Developing a Claims Library. In Proc. IRI '03, 362-369.
105. Perry, D. E., Sim S. E., and Easterbrook, S. (2006). Case studies for software engineers. Tutorial Session in Proc. ICSE '06, 1045-1046.
106. Poppendieck, M., and Poppendieck, T. (2003). Lean Software Development: An Agile Toolkit for Software Development Managers. Addison-Wesley Professional, Boston, MA.
107. Preece, J. Rogers, Y. and Sharp, H. (2002). Interaction Design: Beyond Human-Computer Interaction. John Wiley and Sons. New York, NY.

108. Pressman, S. P. (2005). *Software Engineering. A Practitioner's Approach*. McGraw-Hill, New York, NY.
109. Pyla, P. S. (2007). *Connecting the usability and software engineering life cycles through a communication-fostering software development framework and cross-pollinated computer science courses*. Doctoral Dissertation, Virginia Tech.
110. Pylyshyn, Z. W. (1991). Some remarks on the theory-practice gap. In Carroll, J. M. (Ed.), *Designing Interaction: Psychology at the human-computer interface*. Cambridge University Press, New York.
111. Rosson, M. B. and Carroll, J. M. (2002). *Usability Engineering: Scenario-Based Development of Human-Computer Interaction*, Morgan Kaufman, New York, NY.
112. Salasin, J. (1992) *The Design Record: Keystone of Software Engineering*", Keynote Speech of the Third Reverse Engineering Forum.
113. Sciacchitano, B., Cerwinski, C., Brown, I., Sampat, M., Lee, J. C., and McCrickard, D. S. (2006). *Intelligent Library Navigation using Location-aware Systems*. In Proc. ACMSE '06, 371-376.
114. Schwaber, K. (1995). *Scrum development process*. In Proc. OOPSLA '95, 117-134.
115. Schwaber, K. *Agile Project Management with Scrum*. Microsoft press, Redmond, WA, 2004.
116. Schwaber, K., and Beedle, M. (2001). *Agile Software Development with SCRUM*. Prentice Hall PTR, Upper Saddle River, NJ.
117. Sharp, H., Biddle, R., Gray, P., Miller, L., and Patton, J. (2006). *Agile development: opportunity or fad?*, In proc. extended abstracts CHI '06, 32-35.
118. Sharp, H., Robinson, H.M. and Petre, M. (2009). *The Role of Physical Artefacts in Agile Software Development: two complementary perspectives*. *Interacting with Computers*, 21(1-2) 108-116.
119. Sidky, A. S. (2007). *A Structured Approach to Adopting Agile Practices: The Agile Adoption Framework*. Doctoral Dissertation, Virginia Tech.
120. Sidky, A., and Arthur, J. D. (2007). *A Disciplined Approach to Adopting Agile Practices: The Agile Adoption Framework*. *Agile Journal*. CMC Media, Inc. Available at: <http://www.agilejournal.com/content/view/411/>
121. Soundararajan, S. *Agile Requirements Generation Model: A Soft-structured Approach to Agile Requirements Engineering*. Master's Thesis, Virginia Tech. 2008.
122. Sousa, K. S., and Furtado, E. (2003). *RUPi – A unified process that integrates human-computer interaction and software engineering*. In Proc. ICSE '03, 41-48.
123. Suchman, L. A. (1987). *Plans and situated actions*. Cambridge University Press, New York, NY.
124. Susman, G. I., and Evered, R. D. (1978). *An Assessment of the Scientific Merits of Action Research*. *Administrative Science Quarterly* 23(4), 582-603.
125. Sutcliffe, A. G. (2000). *On the Effective Use and Reuse of HCI Knowledge*. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(2), 197-221.
126. Sutcliffe, A. G. and Carroll, J. M. (2003). *Designing Claims for Reuse in Interactive Systems Design*. *International Journal of Human-Computer Studies*, 50(3), 213-241.
127. Sy, Desiree. *Adapting Usability Investigations for Agile User-Centered Design*. (2007). *Journal of Usability Studies*, 2(3), 112-132.
128. The Standish Group. (1994). *CHAOS*. The Standish Group International, Inc.
129. The Standish Group. (2001). *Extreme CHAOS*. The Standish Group International, Inc.
130. Wahid, S., Allgood, C. F., Chewar, C. M., and McCrickard, D. S. (2004). *Entering the Heart of Design: Relationships for Tracing Claim Evolution*. In Proc. SEKE '04, 167-172.
131. Wahid, S., Branham, S. M., Cairco, L., McCrickard, D. S., and Harrison, S. (2009). *Picking Up Artifacts: Storyboarding as a Gateway to Reuse*. In Proc. INTERACT '09, 528-541.

132. Wahid, S., and McCrickard, D. S., (2006). Claims Maps: Treasure Maps for Scenario-Based Design. In Proc. ED-MEDIA '06, 553-560.
133. Whittaker, S., Terveen, L., and Nardi, B. (2000). Let's Stop Pushing the Envelope and Start Addressing it: A Reference Task Agenda for HCI. HCI 15(2-3), 75-106.
134. Whitworth, E., and Biddle, R. The Social Nature of Agile Teams. In Proc. Agile '07, 26-36.
135. Wixon, D. (2003). Evaluating usability methods: why the current literature fails the practitioner. Interactions 10(4), 28-34.
136. Yin, R. K. (2002). Case Study Research: Design and Methods (3rd Edition). Sage Publications, Inc, Thousand Oaks, CA.

Appendix A. Critical incidents from case studies

Critical Incident Instructions

Study Purpose

I am interested in studying the interactions between usability engineers and other team members including software developers and testers who are working in an agile software development project. I am exploring how team members can collaborate and communicate effectively using various communication channels including face to face meetings, electronic communications and shared design artifacts. I will be observing the development team and the generated design artifacts. In addition, I would like you to record *critical incidents* throughout the development of this project. A *critical incident* is any significant event that either positively or negatively affects the system or behavior that is being studied.

I am interested in critical incidents that either positively or negatively affects *team communication* and *collaboration* for this development project. That is, I would be interested in a critical incident where you are unable to get in touch with an end user to discuss the interface but I would not be interested in an incident where the development database server crashes.

Directions

Although I am mainly interested with events that relate somehow to the usability of the system, I would like you to record any critical incident that significantly affects team collaboration or communication. Please keep the following in mind:

- I would like these incidents to be recorded individually without consulting others.
- These incidents should be written to focus on the *event* more than on any *person or persons*.
- They should be recorded as soon after the event occurs as possible (preferably the day the event occurs but no more than a week after it happens).
- I am interested not only in the types of events that occur but also in the number of times each event occurs. If you record a critical incident event and the same event occurs the following week, then record that as a separate event. It should take approximately 5-10 minutes to record one incident.

Please use the form available here to submit critical incidents. Please record only one incident per form. You can use the link repeatedly to submit other incidents as necessary. As stated in the consent form, all data will be anonymized before being published.

<https://survey.vt.edu/survey/entry.jsp?id=1213021445829>

Example negative and positive critical incidents are included below to give you an idea of the level of detail to include.

Top of Form

Critical Incident Form

Meridium-VT NSF STTR Agile Usability Project

What is your role in the project?

Manager Usability Developer Tester Other:

Date of incident:

Briefly describe the event that either positively or negatively affected team communication and/or collaboration. Please provide any necessary background information related to the event.

I presented the prototype for the search screen in the development planning meeting today. Jake noted that the screen should be returning 'Google'-style results rather than 'Yahoo' style results. I mentioned that I based my design on the design document on the portal. He replied that document I used was out of date.

Why do you believe this positively or negatively affected communication and/or collaboration?

Because I was not aware of where the latest design documents were, my design did not reflect the latest requirements. This negatively affected team collaboration in that the developers will have to wait while I redevelop the screens. This will result in at least a one day delay.

How would you rate the magnitude of the effect (positive or negative) that this incident had with respect to overall development progress?

- 1 - low effect
- 2 - medium effect

3 - high effect

Critical Incident Form

Meridium-VT NSF STTR Agile Usability Project

What is your role in the project?

Manager Usability Developer Tester Other:

Date of incident:

Briefly describe the event that either positively or negatively affected team communication and/or collaboration. Please provide any necessary background information related to the event.

Why do you believe this positively or negatively affected communication and/or collaboration?

How would you rate the magnitude of the effect (positive or negative) that this incident had with respect to overall development progress?

1 - low effect

2 - medium effect

3 - high effect



Self-reported Critical Incidents from Web Components Project.

These incidents were documented by the Web Components team members.

| Role | Date of incident: | Event description | Effect of incident |
|--------------------|-------------------|--|---|
| Usability Engineer | 39486 | I wanted to get started with the Root Concept document and to do so I needed information from the Product Manager. I scheduled a phone interview but he was not able to make it due to another meeting running over. He will be gone for a whole week. | This incident has negatively affected my collaboration with the Product Manager and has resulted in a one week delay in my work. |
| Usability Engineer | 2/18/2008 | I got the Sprint backlog document today in a meeting. The use of terms like 'story' and 'tasks' in software engineering terminology was confusing as the same words mean different things in usability engineering. | This incident negatively affected communication and collaboration as I was initially confused what the words meant in software engineering. |
| Usability Engineer | 2/20/2008 | The development lead sent an email on Monday saying that he wants to start the first iteration on Tuesday. He was supposed to email and schedule a time for the first meeting. It is Thursday now and we still have not heard from him and he has not responded to the email. | This has negatively affected collaboration as I feel left out of the loop. The development team and the lead being in another country does not help. |
| Product Manager | 3/5/2008 | Kick off meeting of sorts. Instead of going into an introduction and purpose we immediately jumped into feature and function review. In this case the development team had already started some prototyping. | Some team members were lost and the team does not have a common level of understanding. Next team meeting will go back and provide the basics to promote a common level of understanding. |
| Other: QA manager | 3/17/2008 | In this morning's stand up meeting I asked a question regarding what has been delivered for iteration 1 for the Web Components and if there is anything documenting that delivery. The answer was no. The last Iteration 1 backlog document is not up to date and there are no worktickets written to describe the delivery. | I believe that this has a negative impact. Dev is working on completing Iteration 2 and QA does not yet know what to test for iteration 1. |
| Other: QA Manager | 3/17/2008 | Development deliveries are being made for Web Components, but not based on Usability Mock Ups | This has a negative impact because it causes lost of confusion and dev and testing rework. |

| | | | |
|--------------------|-----------------|--|--|
| Tester | 3/31/2008 | By the 2nd iteration of testing, no managing tool for the project timelines and dependencies was put in place. | If you have no project management tool to manage delays and dependencies then the project will have higher risks associated with meeting deadlines for everyone. Development, QA, and Documentation organizations will be bombarded with last minute tasks instead of knowing how to plan their work |
| Tester | 4/8/2008 | It occurred to me in another Team Leader meeting that there should be 2-way communication between the Usability Team and all groups involved (dev, qa, prod mgmt, documentation, etc) We need Usability training whereby we become knowledgeable about the underlying theories utilized in coming up with a solution so that we can have a knowledgebase of sorts to refer to in the future. | Teach a man to fish and he can be more self-sufficient...well you know how it goes Prod Mgmt, Dev, and QA should be able to develop projects and come up with "Best Practices" for the customer instead of waiting on 1 particular customer to dictate design. QA will be able to assure that these "Best Practices" are adhered to better. Documentation may see more consistency in code. |
| Usability Engineer | 4/9/2008 | I wanted to look over the web component prototype but I found out that it has not been updated for over a week. | It negatively affected my work as I needed to review the prototype to make usability recommendations. |
| Usability Engineer | 3/26/2008 | At the feature team meeting, there was a long argument about creating work tickets. I had no idea what was going on and time had run out before I could ask. | Negatively affected my communication with the team. |
| Usability Engineer | 4/2/2008 | I was assigned a work ticket but I have no idea what I need to do. | Negatively affected communication because there was no communication in this case, just an email with a work ticket. |
| Usability Engineer | 4/23/2008 | I spent 3 hours writing work tickets on the current prototype and it turns out that the prototype has not been updated. | It negatively affected collaboration and caused extreme frustration over my wasted effort. |
| Usability Engineer | 4/23/2008 | I looked at the iteration plan and could not find any task assigned to me. Seems like the iteration plan is only for the developers. | Negatively affected communication and collaboration as I have been left out of the plan. |
| Developer | April 29th 2008 | I delivered some features on outdev for the Keyword Search functionality. The page was not completely functional and I demoed in the stand up meeting what is working on that screen and what is yet to be delivered. But Usability team look at the screen and was wondering why something was not functional. | There is no way for the developers to convey the details about what is being delivered in outdev to team members who didn't attend the meeting. If someone did not attend one meeting, they will not know what exactly is functional and what is yet to be delivered in a feature. |

Observed Critical Incidents from Web Components Project.

These incidents were observed, documented, and sorted by the researcher.

Collaboration Incidents

| Software Engineer team working ahead of Usability Team | | | |
|---|--------------------------|---|--|
| Role | Date of incident: | Event description | Effect of incident |
| Development Lead, Usability Engineer | 3/5/2008 | Development team did 'development spikes' that ended up including a lot of the UI design. Problem is they have limited understanding of user background/characteristics. UI person proposed getting rid of top control bar in Web Components and integrating it into the sidebar. Dev. team disagreed b/c they thought users would be confused because sidebar controls change depending on the displayed page. Other members of team (Product Manager, <QA manager>) agreed with UI person since controls from top bar would be in common tasks menu on every page like in <Desktop Client>. | Problem is that UI design did not occur far enough ahead of development—resulting in poor UI design and reluctance from developers to rework what they've done. Proposed solution is to have UI work one iteration ahead of developers so we can just hand screens over to them. |
| Usability Engineer | 3/12/2008 | Usability Engineer Says she is busy and won't have much time to work on prototype. She needs access to Meridium client. Problem with usability team and SE team being out of sync. SE team presented more changes to record manager UI, again they did not get feedback from Usability person. (She's at conference) | If the UE and SE teams are not working in sync and the Se team is just working on their own then team is getting no value out of UE team and resulting system may not meet usability goals. |

| | | | |
|-----------------------------|--------------------------|---|---|
| Dev. Team | 3/13/2008 | Developers came up with mockup screens for search functionality and sent them out to the team. Again, they did this without consulting usability team. (And potentially without considering high level usability goals, user classes, etc).) | This can potentially be a problem as evidenced by the issues with<product>. (QA/documentation think there is problem because of inconsistency with existing products. It will be hard to train users to use new system. Developer developed the<product>UI to maximize available vertical space. Problem is they did not properly define and prioritize design/usability goals among the entire team. Also, there is conflict with<product>b/c there was no usability testing done with actual users so there is no certainty as to what will or will not be large usability issues. Different people in team have differing ideas of what's important in terms of UI. This is big problem b/c <PRODUCT> is very near to release date and there is not much time for redevelopment. |
| Developer , Project Manager | 3/14/2008 | At daily meeting, <developer> asked for feedback on what they developed on Friday before Monday so they can get some work done on Monday. Problem is that Usability person is at a conference and may not be able to give feedback by then. Project manager wants to drive develop through mockups. | Risk is that UE and SE teams will remain out of sync. Result will be that the team will not be benefiting from usability. |
| Other: QA Manager | 3/17/2008 | Development deliveries are being made for Web Components, but not based on Usability Mock Ups | This has a negative impact because it causes lost of confusion and dev and testing rework. |
| Product Manager | 3/24/2008 | Product Manager developed the search prototype screens, and <onsite developer> edited it. This was done without first getting input from the usability person. Thus, there may have been limited consideration for high level usability goals and target end users. | This could have consequences for overall usability of the system (overall coherence and consistency). |
| | | Developing UI without consulting Usability | |
| Role | Date of incident: | Event description | Effect of incident |

| | | | |
|--------------------------------------|-----------|--|--|
| QA | 4/1/2008 | Daily meeting with Project Manager, Developers, Tester, QA Manager. Yesterday at the daily meeting Tester noticed a usability problem with the login screen. If you change the datasource and hit enter, it doesn't register. Usually hitting enter anywhere on the login screen will register as hitting the login button. Today, the issue has been fixed. | The daily meetings and incremental dev. process allows for quick turnaround for changes like this. |
| Project Manager | 4/1/2008 | Today at the daily meeting, <developer> demoed the new error message on the login screen. Currently, when there is a login error, a message appears on the screen below the input fields. Typically in the APM product, a popup error message appears. Project Manager had a question of whether Web Components should stick with the onscreen message or be more consistent and go with a popup error message. The usability person was not there and could not provide feedback. | This has low impact but can hinder the type of quick turnaround fixes like the login fix described above. |
| Developer | 4/8/2008 | Developers are working on making datasheets editable. Again, this was done without consulting usability person, but they mostly did this as a proof of concept of the 'editing' concept. Right now you have to double click an individual field to make it editable. Product Manager noted that this is too much action required by user to edit a data record. <developer> countered that it may be hard to make all the fields editable at the same time with the 3rd party ui control they are using. | UE person was again not consulted (she hasn't been available) and no good idea of what developers are working on within an iteration at any specific time. This has a risk of UE and SE remaining out of sync. |
| Usability Engineer, Development Lead | 4/18/2008 | At the daily meeting today, usability engineer was not present. One open item for discussion was whether to do anything about the apparent 'boreness' of the default search page when the user first navigates to it. The developer in charge of implementing the search screens was tasked with developing some UI mockups of different ways to deal with this. | The problem is that this should be the job of the usability engineer, who has a better idea of the target user groups and high level design goals. The fact that the UE was not present is problematic as development must proceed one way or the other. The end result may be a design with overall poor usability. |
| Development | 5/5/2008 | At the daily meeting, <offsite company> demoed work they had done on the catalog and running queries. Product Manager, Project Manager and the team (outside of <offsite company>) was not sure what the functionality was for. Project Manager/Product Manager said he needs to check the requirement/work ticket. This work was done without consulting usability. | UE person was again not consulted.. This has a risk of UE and SE remaining out of sync, and the resulting system not meeting needs of users. |

| | | Development not implementing suggested design decisions from Usability | |
|-----------------------------|--------------------------|--|--|
| Role | Date of incident: | Event description | Effect of incident |
| Usability Engineer | 3/19/2008 | At the weekly meeting, Usability Engineer presented her mockups for the record explorer and generally got good feedback from the rest of the team. But she's not sure if the developers are actually implemented her suggested changes. She mentioned that she has fixes from 2 weeks ago that still haven't been implemented. Issue seems to be that suggested changes are not being folded into the iteration backlog. Something needs to be done to integrate usability fixes into the development backlog. | Risk is that UE and SE teams will remain out of sync. Result will be that the team will not be benefiting from usability. |
| Product Manager | 4/7/2008 | Product Manager has certain small fixes that he has mentioned to developers (at daily meetings) --> for example, the 'run query' link needs to be removed<-- that have not been fixed. There should be an easy way to communicate these fixes to developers without having to write a ticket for them. | Risk is that UE and SE teams will remain out of sync. Result will be that the team will not be benefiting from usability. |
| Product Manager, QA Manager | 4/15/2008 | At daily meeting, Product Manager has a list of minor UI fixes that he came up with based on a meeting with Usability Engineer. He asked how to communicate the fixes to the team so they get developed. QA Manager mentions that there was an issue where these sorts of fixes are mentioned in meetings but are not implemented. She recommends that Product Manager write tickets. Product Manager mentions that it will take 10X longer to write ticket then fix some of these. QA Manager says try to clump them together into tickets. | Problem is that some fixes mentioned in meetings are not being implemented. But writing tickets is time consuming and makes it more likely that some team members won't do them. |
| | | | |
| | | Misunderstandings related to design goals for system | |

| Role | Date of incident: | Event description | Effect of incident |
|-----------------|-------------------|--|--|
| Product Manager | 3/24/2008 | At UI Meeting, Product Manager mentioned that he doesn't like having Developers at the UI meeting because they tend to tell you what's possible and what's not possible. Product Manager doesn't care about whether some is possible or impossible and prefers to 'push' the developers early on. | This is an issue because the prototype may not adequately take into out development issues. Perhaps there should be a developer present to answer questions about feasibility. |
| QA | 3/28/2008 | Product Manager and Usability Engineer did not attend the daily meeting. Tester mentioned issues with the 'common task' task menu. Specifically, the capitalization/wording on the links were not consistent with the APM main product. In addition, Project Manager mentioned that he wants to maintain consistency with the main product as much as possible. This contradicts some of the goals laid out by Product Manager and Usability Engineer. Product Manager stated previously that he is 'open to anything', including breaking consistency with main product if that will increase value/usability of Web Components. There seems to be mental disconnect in the high level goals of the product as understood by Product Manager & Usability Engineer and the rest of the team. | This specific incident is low impact, but this may have more severe ramifications later on--if different members of the team have different priorities in terms of high level design goals, resulting system may not be designed optimally |

| | | | |
|-------------------------------------|--------------------------|---|--|
| Documentation | 4/9/2008 | When seeing the latest prototype demo, Documentation Person mentioned that she thinks the 'about meridium' link should be removed and wants the 'help' link to be displayed at the bottom of the common tasks menu since that would be consistent with the APM product. She mentioned that she realizes consistency is no longer a primary focus of Web Components but does not seem to know what the current high level design goals are. | This can potentially be a problem as evidenced by the issues with <product>. (QA/documentation think there is problem because of inconsistency with existing products. It will be hard to train users to use new system. Developer developed the <PRODUCT> UI to maximize available vertical space. Problem is they did not properly define and prioritize design/usability goals among the entire team. Also, there is conflict with <PRODUCT> b/c there was no usability testing done with actual users so there is no certainty as to what will or will not be large usability issues. Different people in team have differing ideas of what's important in terms of UI. This is big problem b/c <PRODUCT> is very near to release date and there is not much time for redevelopment. |
| | | | |
| | | | |
| | | Design decisions that involve consideration of users and high level design goals | |
| Role | Date of incident: | Event description | Effect of incident |
| Product Manager, Usability Engineer | 3/18/2008 | As the weekly meeting (hourly meeting every Wednesday) tends to focus on development issues and demoing the system, Product Manager and the usability person and the lead developer scheduled a separate weekly meeting to focus on UI issues (e.g. UI design, requirements, evaluation plans). This meeting focused on UI issues related to the record explorer and the upcoming search functionality planned to start development in the next iteration. Also discussed an upcoming usability evaluation with customer planned in the next week to evaluate what has currently been implemented (screen framework, explorer). | This will help focus more on usability issues and concerns and how it fits into overall product plan. |

| | | | |
|--|-----------|---|--|
| Usability Engineer, Documentation | 4/23/2008 | At the feature team meeting, Project Manager brings up the issue of whether fields that users can't edit should be disabled or whether it should work as it does in APM. Documentation Person mentions that if fields are disabled the functionality will be different from APM and this might confuse existing customers. Usability Engineer mentions that most of users will be novices who do not have much experience with existing project. In addition, disabling fields would be more usable solution. | This shows how proper definition and understanding of end users can justify design changes that go against past design decisions. |
| Documentation, Project manager, Usability engineer | 4/23/2008 | At the feature team meeting, Project Manager brings up the issue of whether fields that users can't edit should be disabled or whether it should work as it does in APM. Documentation Person mentions that if fields are disabled the functionality will be different from APM and this might confuse existing customers. Usability Engineer mentions that most of users will be novices who do not have much experience with existing project. Disabling fields would be more usable solution. But Documentation Person mentions that this may not be possible b/c of underlying architecture limitations. Need to meet with Architecture Person to discuss architecture and whether change is possible | This shows conflicts that can arise between developers and usability. Proper resolution of conflict will depend on severity of problem, importance of problem and cost to fix the problem. |
| Product Manager, Usability Engineer | 4/28/2008 | <UE> asked why there is a requirement to have 'multiple home pages'. It doesn't make sense from a usability standpoint. Product Manager says it's because APM supports the concept of multiple home pages so web components needs to support it. He admits that this will be a seldom used feature. | Product Manager gave reasoning that conflicts with high level usability goals. This is reducing usability of overall product for little benefit. Usability Engineer did not object against this. This may be communication problem |

Communication Incidents

| Role | Date of incident: | Event description | Effect of incident |
|------|-------------------|--|--------------------|
| | | Successful communication between team members | |

| | | | |
|-------------------------------|--------------------------|--|---|
| Project Manager | 3/19/2008 | At weekly meeting tomorrow, Project Manager wants to have an official 'kick-off' meeting to go over the development process being used and to review high level design goals and vision for the new web components. Development has been moving forward for a month and the team is only now coming together (QA, documentation). There is a need to get everyone on the same page with regard to the project. | This meeting should have been done at the start of the project but the entire team could not be put together at the beginning. Could affect high level project design efforts. |
| QA | 3/24/2008 | At the daily meeting, Tester asked a question asking what QA should be testing against. Currently, there are lofi mockups, a prototype on outdev (usu. one or two days ahead of delivered work), and the delivered work. Product Manager said that she should test against delivered work, but if she sees problems in the lofi mockups or prototype, then she should communicate those to the team informally (i.e. not through ticket system). | Blocking issue resolved in daily meeting. |
| Developer, Usability Engineer | 4/11/2008 | <onsite developer> had a question regarding number of results per page that should be returned for search. As Usability Engineer was not in the office, she sent an email out to Product Manager and Usability Engineer to get feedback. The issue was able to be resolved over email. | Since Usability Engineer mentioned in the daily meeting that developers should email her if she's not around and there are questions about usability, <onsite developer> did just that. This will help collaboration through use of alternative communication channels when all team members are not physically present. But should this design decision be recorded somehow? |
| | | | |
| | | Communication failure due to limited involvement from team member | |
| Role | Date of incident: | Event description | Effect of incident |
| Documentation? | 3/10/2008 | At daily meeting. Not everyone is attending the daily meetings. Although developers, QA, product managers and the team leads at least have one representative there, there is currently no one from documentation attending these meetings. | This could be problematic later on in terms of getting the documentation person up to speed. |

| | | | |
|-------------------------------------|--------------------------|---|--|
| QA | 3/24/2008 | At daily meeting, Tester mentioned that she has no clear sense of the iterations (project progress) from a QA/documentation perspective. They are not able to attend every daily meeting so can lose sense of where the team is in the project. | Could be problem when testing functionality and writing documentation later on. |
| QA Manager | 4/16/2008 | At feature team meeting, QA Manager asked what the relationship was between the high level plan & the iteration backlog. Project Manager explained that the high level plan contains features that will be implemented (roughly) for each iteration. The backlog contains detailed implementation info for each iteration. | Problem was that there was misunderstanding among some team members (QA, doc) as to how project materials are being used to track progress. Now that issue has been explained, hopefully will allow team members to better track project progress. |
| | | | |
| | | Team member fails to respond to another team member's communication | |
| Role | Date of incident: | Event description | Effect of incident |
| Usability Engineer | 2/8/2008 | I wanted to get started with the Root Concept document and to do so I needed information from the Product Manager. I scheduled a phone interview but he was not able to make it due to another meeting running over. He will be gone for a whole week. | This incident has negatively affected my collaboration with the Product Manager and has resulted in a one week delay in my work. |
| Usability Engineer, Product Manager | 2/13/2008 | Phone conference call with Development Lead, Developer and QA people at <offsite company> in <loc>, and Usability Engineer and Myself in Blacksburg Office. We would like to get in touch with primary customer contact (Product Manager) to discuss project vision, user roles and high level goals for the web components project. He is out of town on a business trip (Bahrain) so we are unable to meet with him face to face. We sent an email to him requesting information but he has not responded (3 days now). | This is hindering communication & collaboration in that there is a significant delay between messages from us to him so we are unable to move forward in terms of the UI design. |
| Usability Engineer | 2/20/2008 | The development lead sent an email on Monday saying that he wants to start the first iteration on Tuesday. He was supposed to email and schedule a time for the first meeting. It is Thursday now and we still have not heard from him and he has not responded to the email. | This has negatively affected collaboration as I feel left out of the loop. The development team and the lead being in another country does not help. |

| | | | |
|-------------|--------------------------|--|--|
| | | | |
| | | Team member fails to initiate communication with another team member. | |
| Role | Date of incident: | Event description | Effect of incident |
| Developer | 4/11/2008 | In the daily meeting development said they were confused about something from the mockups. They weren't sure whether there should be control buttons both on the top of the sheet and on the left nav bar. Was not clear from the mockups which way to go. Developer wanted to clear this up with Product Manager. Usability Engineer told him (and the team) to send her & Product Manager an email if such issues arise. | If developers are confused about some aspect of the UE mockups or other artifacts and they make no effort to resolve the inconsistency in a timely manner, this could cause/exacerbate the rift between UE and SE. Usability Engineer proposed that if similar problems come up to just email her. |
| Development | 4/24/2008 | Developer has been assigned a number of tickets. He says he needs more info (e.g. mockups) on some of these tickets, esp. the usability tickets. But he hasn't contacted Usability Engineer or Product Manager about getting more info. Project Manager and Tester say to email Usability Engineer or Product Manager if this happens again. | This shows that the team is not using proper communication channels to resolve issues that come up. This is hindering collaboration and delaying development. |
| | | | |
| | | Lack of common ground between team members. | |
| Role | Date of incident: | Event description | Effect of incident |

| | | | |
|--------------------------------------|-----------|--|--|
| Usability Engineer, Development Lead | 2/13/2008 | Phone conference call with Development Lead, Developer and QA people at <offsite company> in <loc>, and Usability Engineer and Myself in Blacksburg Office. Development Lead does not seem to have good understanding of role of Usability Engineer. (just think they design UI screens). Doesn't understand other things we do in terms of user/tasks analysis, usability evaluations, etc. Seems to be disconnect between what he wants to do and what we want to do. We want something like a root concept document to high level requirements. Development Lead wants us to come up with specific ideas for screens. | I believe this disconnect is resulting in communication problems between usability and software developers. |
| Development Lead | 2/15/2008 | Development Lead sent a preliminary project backlog through email. Problem in that it is very system-centric and detailed and makes assumptions about implementation. | This is hindering communication and collaboration in that we seem to be thinking about the system at different levels. We don't share a common vision about what the system will do and what the core features are. Set up a phone conference with Development Lead and Usability Engineer to discuss these issues and try to come up with some common ground. |
| Usability Engineer | 2/18/2008 | I got the Sprint backlog document today in a meeting. The use of terms like 'story' and 'tasks' in software engineering terminology was confusing as the same words mean different things in usability engineering. | This incident negatively affected communication and collaboration as I was initially confused what the words meant in software engineering. |
| QA Manager | 3/5/2008 | There is problem with different terminology. Dev team previously used SCRUM process and are using similar process/artifacts/terminology. Meridium is currently using process/terminology from MSF Agile framework. <management> does not want two separate processes because it will only confuse other people on team. <QA manager> (tester) was confused by the process and use of terms (didn't understand that 'release backlog' was the set of all features that are going to be delivered.--> didn't know difference between that and the 'iteration backlog'). | Miscommunication can confuse team members who join late or who are less involved in the development effort (e.g. documentation) |
| Product Manager | 3/5/2008 | Kick off meeting of sorts. Instead of going into an introduction and purpose we immediately jumped into feature and function review. In this case the development team had already started some prototyping. | Some team members were lost and the team does not have a common level of understanding. Next team meeting will go back and provide the basics to promote a common level of understanding. |

| | | | |
|--------------------------|-----------|---|---|
| QA | 3/24/2008 | At daily meeting, there was a confusing use of terminology. Tester used the word 'mockup' to refer to the functioning prototype on outdev. This was confusing as she was trying to discuss how the testing process would go and the rest of the team uses 'mockup' to refer to the nonfunctional powerpoint slides. I questioned her on the use of terminology and corrected her. | Miscommunication can confuse team members who join late or who are less involved in the development effort (e.g. documentation) |
| Usability Engineer | 3/26/2008 | At the feature team meeting, there was a long argument about creating work tickets. I had no idea what was going on and time had run out before I could ask. | Negatively affected my communication with the team. |
| Development Lead, Tester | 4/9/2008 | While discussing an issue with the project Development Lead used the term 'sprint'. QA Manager immediately corrected him and told him to use 'iteration'. | Miscommunication can confuse team members who join late or who are less involved in the development effort (e.g. documentation). --Although this incident shows how team members are beginning to standardize terminology |
| Developer | 4/15/2008 | At the start of the daily meeting, <developer> used the word 'retrospective' to refer to reviewing the last iteration's work. This is different from how I and Development Lead use the word retrospective in an agile sense. | This type of misuse of terminology shows that there are still issues within the team with developing common ground. This can result in misunderstandings and problems within the team. |
| Project Manager | 4/30/2008 | Non-functional requirements (e.g. latency, # concurrent users and other performance reqs) have not been defined yet. Previous version of Web Comp. also did not have nonfunc. Requirements. | If these had been defined at the beginning, they may have impacted the way the system was implemented and affected the design. Defining them this late may result in extensive rework to meet them. |

Information Sharing Incidents

| | | | |
|-------------|--------------------------|--|---------------------------|
| | | Team members do not have easy access to latest implementation | |
| Role | Date of incident: | Event description | Effect of incident |

| | | | |
|-------------------------------------|-----------|--|--|
| Dev. Team | 3/10/2008 | At daily meeting. Development team made current functioning prototype available to other team members through an internal website (outdev). This was not accessible to other team members because the application server was down. This issue was resolved the following day. | If the entire team does not always have access to the latest prototype, then team progress can slow down if they have to wait on developers. |
| Dev. Team | 3/26/2008 | Usability meeting with Product Manager and Usability Engineer Developers started integrating catalog items but didn't upload their latest work to outdev so Product Manager couldn't show Usability Engineer What has been worked on. (Usability Engineer could not attend the daily meeting that day). | This is a relatively low impact problem but not regularly updating outdev will result in team members not having access to the latest prototype being worked on. |
| Usability | 4/8/2008 | Usability Engineer wanted to review the latest updates to the prototype but could not access it b/c latest code hasn't been uploaded to outdev. | This is a relatively low impact problem but not regularly updating outdev will result in team members not having access to the latest prototype being worked on. |
| Product Manager, Usability Engineer | 4/11/2008 | In UI meeting, Product Manager wants to show Usability Engineer how datasheets have been implemented but the outdev copy hasn't been updated in over a week. Usability Engineer was told Project Manager is working on getting a new machine for that. Product Manager: ""I wish I could access this from outdev". | This is hindering collaboration as Product Manager and Usability Engineer are unable to look at and ref. the latest work of the developers to provide UI guidance and feedback. In addition Usability Engineer has not emailed the developers or otherwise contacted them to get this needed information. If the team is not communicating with each other then progress will slow down and inconsistencies and misunderstandings will emerge. |
| Development Lead, Product Manager | 4/15/2008 | At daily meeting, Development Lead says that outdev will be updated on a daily basis from now on. | This will make it possible for team members (including usability) to use the latest implementation and get a sense of how it works. |
| Usability Engineer | 4/23/2008 | Usability Engineer starts writing work tickets for usability fixes, then realizes that outdev hasn't been updated since last week. She has wasted her time writing tickets. | Usability Engineer does not have a good understanding of project progress (what has been implemented and when) as her daily meeting attendance is sporadic. In addition, the outdev prototype is supposed to be updated daily but again has not been updated for a while. |

| | | | |
|-------------------------------------|-----------|--|---|
| Usability engineer | 4/23/2008 | Usability Engineer cannot see latest work developers have done because outdev hasn't been updated since last week. Tester teaches Usability Engineer how to install latest RCs which are delivered every week. | For colocated teams, team members can walk over and look at others' screen to see work in progress. This is not possible with this distributed team so outdev is the best substitute we have. It needs to be updated so everyone maintains awareness of progress within an iteration. |
| Project Manager, Development | 4/24/2008 | Project Manager mentions that outdev does not have to be stable, but it should be very current. Now it does not appear to be current. Developer says it was updated up to next monday but they are still developing so they haven't updated since then. Project Manager says don't worry if it's working perfectly, it's more important that it's updated. | For colocated teams, team members can walk over and look at others' screen to see work in progress. This is not possible with this distributed team so outdev is the best substitute we have. It needs to be updated so everyone maintains awareness of progress within an iteration. |
| Usability engineer | 4/28/2008 | Usability Engineer is waiting for outdev to be updated so she can run through it and identify tasks for the usability evaluations on Wednesday. She is hesitant to just install the latest RC b/c she is not sure if it is the latest version of web components. She is still not sure when RCs are delivered. | This illustrates a communication problem between usability and development--and usability person has made no effort to resolve this miscommunication. |
| Development | 4/28/2008 | <onsite developer> has updated outdev with her latest work on keyword search. She is concerned because not all of the keyword search functionality is working and she is not sure how to communicate to the rest of the team how much of it is done. | This is a collaboration/communication issue as the rest of the team members (several of whom are distributed) may not be sure what is and is not completed with respect to keyword searching. |
| Product Manager, Usability Engineer | 5/8/2008 | At usability meeting: While reviewing the latest implementation on outdev, they noticed that the new design for the results grid that <onsite developer> worked on has been overwritten. Seems like when <offsite company> developers were updating it, they overwrote her changes. | Another consequence of a distributed team. In this case, one of the developers is in roanoke and the rest are at <offsite company>. Poor communication can cause issues like this which affect other team members who need outdev to review latest work. |
| | | | |
| | | | |

| | | | |
|--|--------------------------|---|--|
| | | Failure to share design documents between team members | |
| Role | Date of incident: | Event description | Effect of incident |
| Product Manager, Usability Engineering | 2/6/2008 | During planning meeting, we asked Product Manager if it was possible to get access to existing online system from Company X to get understanding of how it works. Product Manager says we don't have access from Meridium. Can only get access to screenshots. | This is hindering development progress in that we will not have as good an understanding of how the existing system works and why certain design decisions were made |
| Project Manager | 3/10/2008 | At daily meeting. This issue was brought up by Project Manager and discussed by the team (Myself, Project Manager, <onsite developer>). Problem of sharing design documents and information between distributed team members. Both <offsite company> and Meridium have their respective Portal systems for sharing documents but these two systems are incompatible. Project Manager asked Developer if he has latest iteration document. We are currently using email to share docs., but problem is that documents get out of sync. <onsite developer> is tasked with looking into alternatives for document and information sharing. | This can hinder information sharing and potentially cause confusion if someone does work based on outdated design documents/information. |
| | | Sharing information through portal accessible to all team members (remote and local) | |
| Role | Date of incident: | Event description | Effect of incident |

| | | | |
|--------------------------------|-----------|--|---|
| Project Manager | 3/10/2008 | Project Manager met me at my desk. He mentioned he has uploaded the daily meeting notes to the portal and said that the portal should now be accessible to everyone (including people at <offsite company>). His concern is that the UI/layout of the Portal is not user friendly and makes it hard to find and keep track of information. | This should make it easier for people in <offsite company> and Meridium to share information |
| Project Manager | 3/19/2008 | At daily meeting, Project Manager mentioned he wants to start using the Portal more as a shared workspace since the team is distributed between Here and <loc>. As there have been problems with keeping track of documents and meeting notes, etc he wants to use the Portal as a virtual workspace. This will include announcements, agendas, work documents, etc. (but is team accessing and using it?) | This should make it easier for people in <offsite company> and Meridium to share & disseminate project information. |
| Project Manager | 4/16/2008 | The project manager wants to increase overall team awareness of project progress and remain aware of open issues. The goal of the PM is to use the Portal site as a common point through which the team can achieve this. To this end, the portal site was revamped to show day to day items (discussion items, action items) at the top and will not be regularly maintained by the PM. | This will have a positive impact on overall development progress by increasing shared understanding of the project among team members, especially among QA and documentation. In addition, this will prevent proposed work items and discussion items from daily meetings from 'falling through the cracks' |
| Project Manager | 4/18/2008 | In the daily meeting, the project manager reviewed all open discussion items and action items currently on the portal with the rest of the team. | Regular usage and exposure to the portal site and how it will now be used will increase the likelihood that other team members will adopt and 'buy into' it |
| Project Manager, Documentation | 4/23/2008 | At the feature team meeting, Project Manager explains briefly to Documentation Person how the Portal works, and describes the flow of information from discussion items to action items or iteration modification requests to work tickets (iteration modification and request has been moved out of excel sheet and onto the portal). | This goes further towards having the entire team understand how the project is being managed and will help to maintain a common understanding of process and project progress |
| | | | |
| | | Improper use of project tracking software to share information between team members (problem b/c not all team members are involved on a regular basis) | |

| Role | Date of incident: | Event description | Effect of incident |
|-----------------------------|-------------------|--|---|
| Other: QA manager | 3/17/2008 | In this morning's stand up meeting I asked a question regarding what has been delivered for iteration 1 for the Web Components and if there is anything documenting that delivery. The answer was no. The last Iteration 1 backlog document is not up to date and there are no worktickets written to describe the delivery. | I believe that this has a negative impact. Dev is working on completing Iteration 2 and QA does not yet know what to test for iteration 1. |
| Usability Engineer | 4/2/2008 | I was assigned a work ticket but I have no idea what I need to do. | Negatively affected communication because there was no communication in this case, just an email with a work ticket. |
| QA Manager | 4/8/2008 | QA Manager noted that documentation isn't here so we need to document stuff in tickets so we can communicate stuff to them at later point. Documentation dept. has limited resources and is not committing anyone right now. | Not having documentation person as part of team will mean they will have problem doing their job later on. Much of discussions/context will be missed. Writing some information in tickets will mitigate some of these issues. |
| Tester | 4/11/2008 | At the daily meeting, Tester wants to know how we are keeping track of usability fixes/changes/additions (e.g. has something been fixed? Whose responsibility is it, if not fixed, when will it be fixed, etc). She is not seeing things change/update on the portal. | Problem is that team is currently using number of different ways to track work tasks (work ticket system, iteration backlog, action items, 'iteration & recommendation & modification excel doc). Not all of these are updated regularly and it's unclear where to look for what changes. This is hampering collaboration and communication as it is hard for some team members to track project progress |
| Project manager, QA manager | 4/23/2008 | At feature team meeting, QA Manager mentions that the list of open tickets is getting very large. They need to be updated and completed tickets need to be closed. Closed tickets are a commonly used measure of progress and right now it does not appear that Web Components is making progress. | Ticket's are primary mechanism through which QA and documentation track project data and project progress. If it's not updated regularly, they will not be able to work effectively. |

| | | | |
|--------------------------------|--------------------------|---|--|
| Project manager, Documentation | 4/23/2008 | At feature team meeting, Documentation Person mentions that tickets currently do not have much detail. She is concerned because tickets are the primary mechanism through which she gets information about projects she is involved in. | There is still some disconnect about how project information is being recorded and tracked. Documentation Person wants ticketing system to be primary documentation mechanism and Project Manager is trying to use the Portal as the primary project management mechanism with ticketing system also used to track project progress. |
| Usability | 5/8/2008 | At usability meeting: There are a large number of small UI changes from iteration to iteration. In this case, changes based on last week's evaluation and on a review of the latest implementation on outdev today. Currently, usability person has to write work tickets for each change. | Writing and managing work tickets is very time consuming. There should be a more efficient way to communicate some of these changes to developers. |
| | | Using multiple artifacts to track project progress causes confusions about project progress | |
| Role | Date of incident: | Event description | Effect of incident |
| Development Lead | 3/5/2008 | From Wed. meeting. There is currently no plan in place for integrating the Scrum iteration plan that Development Lead put together with the work ticket system that is already in place and that the QA and documentation folks depend on. In addition, there is no way to integrate UI tasks with either process (either the Scrum plan or the Meridium work ticket system). Proposed solution is to use integrated approach using MS Team system. | This could potentially hinder collaboration between development team and QA/documentation teams because QA/doc depend on ticketing system to track changes that need to be made. |
| QA Manager | 3/31/2008 | Project Manager brought up the excel sheet showing what features were delivered in iteration 3. QA Manager noticed that certain items were not completed. Project Manager mentioned these items will be moved to the next iteration. QA Manager asked what impact this will have on the overall project. Without some type of project plan, they don't know. | QA Manager reiterated that without an estimate of 'remaining work', if something slips we don't know what the impact will be on the overall effort. Maybe have/maintain a release-level 'burndown chart' to measure relative progress. |

| | | | |
|-------------------------------------|-----------|---|---|
| Product Manager, Usability Engineer | 4/11/2008 | Usability Engineer had Product Manager open up the 'Iteration Recommendation & Modification' excel document to review some of the recommended changes and open issues from the developers. Product Manager admits that this is the first time he's seen this document. Usability Engineer also notes that it's unclear who added what item and why. | Problem is that team is currently using number of different ways to track work tasks (work ticket system, iteration backlog, action items, 'iteration & recommendation & modification excel doc). Not all of these are updated regularly and it's unclear where to look for what changes. This is hampering collaboration and communication as it is hard for some team members to track project progress |
| QA manager | 4/16/2008 | At feature team meeting, QA Manager asked what the relationship was between the high level plan & the iteration backlog. Project Manager explained that the high level plan contains features that will be implemented (roughly) for each iteration. The backlog contains detailed implementation info for each iteration. Problem is that there are discrepancies between the high level plan and the iteration backlogs (e.g. wording differs between features described in the two). | If project documents are not updated regularly or if there are discrepancies between different project documents, then team members may lose faith in the usefulness of the portal as a window into project progress. Also, this increases confusion and could hinder project progress |
| QA manager, Project Manager | 4/16/2008 | At feature team meeting, when shown the high level plan, QA Manager asked how dependencies between features are not shown. If a feature slips, feature complete could be in jeopardy because dependent features would also slip. <--and we wouldn't know because we don't know what dependencies are. | This could have a potentially huge impact on project progress because this risk is unknown and could push back the delivery date. Project Manager suggests using MS Project to create a minimum project plan so dependencies can be tracked. |
| Usability Engineer | 4/23/2008 | Usability Engineer is not sure what to work on. Her tasks are not included in the Iteration Backlog any more. Only development tasks are tracked. | It's not clear how usability tasks are assigned and managed. This will affect project progress evaluation and project planning. |
| Project Manager, QA | 4/24/2008 | Tester was confused b/c a ticket related to administration was not delivered but iteration plan says it was completed (iteration 6). Project Manager says she needs to look at 'load' field in the ticket system. He went through ticket system with Development Lead to match 'load' field with the iteration. She says need to make sure if an iteration ends, all done tickets are closed. | Discrepancies between ticketing system and high level plan on the portal will confuse team members (esp. QA and doc) about what has and hasn't been done with the project. |

| | | | |
|------------------------------|-----------|---|--|
| Project Manager, QA manager | 4/30/2008 | There are a large number of open work tickets related to fixes/changes for already delivered features. The number of such tickets is growing with each iteration. | QA Manager mentioned that it will be better to have some test&fix at each iteration rather than push it all to the end. It's easier to fix if feature is fresh in developers mind. Also, this will ensure that later dependencies/issues don't affect development later on. This would be the more agile way to do things. |
| Project Manager, Development | 5/7/2008 | At feature team meeting: Project Manager is doing much of his project planning using the work tickets. He was trying to update the project plan on the portal but found that many of the tickets haven't been updated. | Developers not updating work tickets is problematic both for project planning and it affects QA and documentation because ticket states are used to determine what gets sent over to QA and documentation so they can get their work done. |
| QA Manager | 5/19/2008 | At the daily meeting, Project Manager showed the project plan (in MS Project). <QA manager> mentioned that she has not looked at the project plan because she finds it hard to follow. She doesn't understand how to tell if a task is done or has been moved. Project Manager has a particular way of managing the plan that she doesn't understand. | Communication issue that relates to high level view of project progress. <QA manager> suggests that Project Manager write a document explaining how he is managing project progress. Project is almost over so it may not matter as much. |

Technical issue incidents

| System failure related to development machines | | | |
|---|--------------------------|---|---|
| Role | Date of incident: | Event description | Effect of incident |
| QA Manager | 3/31/2008 | Daily meeting (Product Manager and Usability Engineer could not attend). Development has been using a QA app server b/c that is the only machine that has a working <REPORT> server. Development is unable to get <REPORT> working on a dev. machine though they are working on it. QA Manager expressed concern—can't have people outside QA messing with a QA server. | If something happens with the server, then won't know who did what. QA Manager wants this to be fixed as soon as possible. Currently this does not have a significant impact on progress though it could cause serious problems later on. |

| | | | |
|--------------------|--------------------------|--|---|
| Usability Engineer | 4/30/2008 | Usability engineer planed to run brief usability evaluations with employees at meridium to get feedback from new users. During the first test, outdev failed so the evaluation was delayed until it was brought back onlin about an hour later. | Technology failure that delayed evaluation. |
| Project manager | 5/20/2008 | At the daily meeting, Project Manager wanted to show performance requirements written into a ticket in <work ticket system>. He could not show it right away because there were no free licenses. | This is a minor technical issue that hindered communication with other team members. QA Manager went to her desk and logged out of <work ticket system> so Project Manager could get a license. |
| | | | |
| | | | |
| | | Communication system/software failure | |
| Role | Date of incident: | Event description | Effect of incident |
| Development | 4/24/2008 | Couldn't do demo today b/c Developer couldn't get a remote meeting setup (no id is free). Project Manager did not want to try to do it on his side as it takes too long to setup. They are unable to demo their work. <onsite developer> was unable to demo the mockups. | Remote meeting software is primary way of demoing work at daily meetings for distributed team. Not having access to it will hinder communication of work progress. This is an infrequent occurrence |
| Development | 5/5/2008 | <developer> sent a goto meeting request for the daily meeting today but no one except Project Manager got the invitation. | Technology failure that delayed meeting. |
| Developer | 5/9/2008 | At daily meeting, Developer was unable to schedule remote goto meeting to demo their latest work (there were no available IDs). | Technical glitch that prevented development team from demoing their work hinders communication (even though team can look at work at later time). |
| Project manager | 5/12/2008 | At daily meeting, developers were demoing latest work through goto meeting. Halfway through demo, there was a network problem on the developers side and the goto meeting failed. Had to stop the rest of the demo. Work can still be reviewed by team later through outdev. | Technical glitch that prevented development team from demoing their work hinders communication (even though team can look at work at later time). |
| Project manager | 5/12/2008 | Before daily meeting, Project Manager sent an email saying that a conference line is unavailable so developers should directly call the conference room where meeting is held. | Minor technical glitch easily resolved through email. |

| | | | |
|------------|-----------|--|---|
| QA Manager | 5/19/2008 | There was a meeting related to Search today but QA Manager did not get a reminder from her Outlook/Exchange. She mentions that she has had problems with not getting meeting reminders consistently. Project Manager mentions that he also has problems with the calendar system--not being about to see his own calendar. | This is a minor technical issue that hinders collaboration. |
|------------|-----------|--|---|

External influence incidents

| Role | Date of incident: | Event description | Effect of incident |
|----------------------------|-------------------|--|--|
| | | External influences that hindered progress - External project dependencies and influences that affected project progress | |
| Development | 4/29/2008 | Work on Keyword Search feature is slipping week to week b/c they are unable to continue development until <core dev 1>or <core dev 2> (Core Developers outside of Web Components Team) get core keyword search functionality implemented. | Dependencies with projects outside of Web Components have the risk of holding up development of Web Components. |
| Project Manager | 5/8/2008 | At feature team meeting, PM mentioned that original Feature Complete date was <date>. Project manager wanted to get to feature complete one month early (<date>), so team has time for comprehensive regression testing. Project is slipping so it will not only be 1 week early. Key problem is external dependency keyword search core work isn't done yet because all resources committed to getting <version> out. | Dependency external to project has risk of further pushing back feature complete date and affecting rest of development/test schedule. |
| Project manager, Developer | 5/9/2008 | The daily meeting was just a phone conference because the <product> team requested meeting room from Project Manager. | This hinders communication in that team members in Roanoke were not in the same physical space (resource limitation) |
| Developers | 5/19/2008 | Feature complete will be pushed out--> team is still waiting on core work to get keyword search working. Also, there is new requirement to add configured explorer functionality to web components. | Dependency on developers outside the team has been a limiting factor throughout the project. |

Self-reported Critical Incidents from Touch Screen Project.

These incidents were documented by the Touch Screen project team members.

| Role | Date of incident: | Event description | Effect of incident |
|--------------------|-------------------|--|--|
| Usability Engineer | 6/11/2008 | This event occurred during the 30-minute (run over to 60-minute) stand-up meeting. Because of the inability to work directly with the customer, I proposed the idea to compose a customer questionnaire. I was surprised by the initial acceptance (by <qa manager>) and eventual acceptance (by <project manager>, and perhaps <documentation mgr>) to move forward with this idea. This is an example of positive communication/collaboration. | This event is positive in two ways. First, it shows that even the QA and Documentation team members are sensitive to usability concerns (they seem to be the most resistant in other situations). Second, it shows a willingness of the team to discuss and debate matters with an open mind and to reach new conclusions given dynamic information. |
| Usability Engineer | 10/13/2008 | This event occurred in a 30-minute stand-up meeting. <project manager> introduced a requirements meeting to be held with <product manager>, and I asked if I could be present at the meeting as well. I was happy to find <project manager> willing to include me (and then another individual). This was a positive collaboration event. | One point of worry is that the team lead initially did not consider the usability engineer a vital participant of the meeting. However, the event is, on the whole, positive because it shows that the team lead is willing to collaborate with the usability engineer. |
| Project Lead | 6/18/2008 | <product manager> has been unavailable for about a week to talk with. He hasn't been at daily meetings, available for specific meetings, or even answer direct email questions. | no comm |

| | | | |
|--------------------|-----------|---|--|
| Usability Engineer | 6/16/2008 | <product manager> has been absent from the team, making it difficult to access the user. He hasn't responded to any of the emails I've sent him and he's missed the last 3 stand-up meetings. The particular event I am documenting here began on the 16th when I sent <product manager> an email with some customer questions to send to <customer representative>, the <customer company> representative. Two days later, after no response from <product manager>, I resent him the email with an update and a request that he send it out before close of business. I received no response. I finally received an indirect reply on our conference call when <product manager> said something to the effect of: "<usability engineer>'s already got a good list of questions that she can ask once we get to <customer company>." I suppose my questions will have to wait another week.... | This event shows a breakdown in communication as well as collaboration. Communication became unidirectional, which disallows for any feedback or compromise. Also, <product manager> seemed unwilling (or simply too busy) to work with me to help me get access to the information I need--a blow to the sense of team collaboration. |
| Usability Engineer | 6/20/2008 | I received a forwarded email today from <product manager>. It's the first real insight I have to the <customer company>-Meridium conversation. I was pleased to find the <product manager> had forwarded on my site visit requests to <customer representative> without modification. | This event is positive in that it shows that <product manager> is making some effort to maintain communication and work with other teammates to help them get what they need. |
| Usability Engineer | 6/20/2008 | Today in our standup meeting, <qa manager> and <developer>/<developer 2> had a bit of a debate over which version of the APM to build our product around. I think there were collaborative tensions between the QA and Dev groups on this issue that should be worked through in a meeting outside of our standup. I was discouraged that a meeting wasn't proposed to help bring all concerned parties to the table to make an informed decision. | I think this was a negative incident. There should have been a call for a separate meeting to discuss this issue because no resolution was made and neither side backed down. I'm worried that a decision will be made by development that doesn't consider QA concerns and will hurt our team unity. |
| Usability Engineer | 6/20/2008 | Last Monday, <qa person> did some background research on touchscreen prices of his own volition. I decided to send him a "thank you" email today and I got a really great response. He said it was his pleasure to do the research and asked me if I had anything else for him to lookup. | This is a positive event that shows the devotion of a team member to the project and his desire to help make contributions that are meaningful to other team mates. |

| | | | |
|--------------------|-----------|--|--|
| Other: QA Manager | 6/20/2008 | In today's stand up meeting there was a discussion on use of <integration product>. The discussion was regarding whether to use <> released product or use a <>non-released RC version of the connector. | I think that the discussion and up front communication of this decision in the stand up will have a positive impact on the project. In the past, before feature teams and stand up meetings, it is possible that the developmnet decision may have been made without much discussion and planning. Regardless of what the decision is, the discussion, risk assessment and planning that should be the outcome of these types of early disucssions will go a long way towards risk mitigation and possible customer impact. It will be important to have further discussions on this topic |
| Usability Engineer | 6/23/2008 | I saw <product manager> in the office when the team had thought him to be away on business. He also didn't attend our team meeting. I'm surprised he didn't tell us he was still in town, given that we worked so hard to schedule a meeting with him before Friday (when we thought he was leaving). | I think this is a negative incident that shows lack of team communication. With a business trip to the customer site coming up, I would have expected <product manager> to be at our meeting and to at least let us know his travel plans and when they change. |
| Developer | 6/20/2008 | The discussion on which of Meridium to use is causing concerns. <newer> will give us better security because we don't have to hard code the customer's <> password in to the system, but QA insists on <older>, just because it's already a released product. And I don't think that's a good enough reason. | The negatively effects the communication, because I feel like QA is "no" a lot, without substantiating their concerns. I think we can take some calculated risks, so we don't have to rewrite the code in Phase 2, when they discover <newer> is actually more secure. This I think slows down communication. |
| Usability Engineer | 6/23/2008 | <project manager> didn't realize that I had uploaded several documents to a new <customer company> Visit 1 folder on the team portal. I was surprised by this because I wrote an email last Friday notifying the team about this new folder and contained documents. The email contained a link to the document. | I think this was a negative event because it involved communication breakdown. Because <project manager> was unaware of the document contents, he focused our meeting on the old documents; perhaps we got less relevant feedback from the team because of this. Also, he scheduled a meeting later in the day to cover the contents of the new documents. This meeting might have been much shorter if he had reviewed the new documents in time for our team meeting. |

| | | | |
|--------------------|-----------|---|--|
| Usability Engineer | 6/23/2008 | <project manager>, <researcher>, and I had a meeting today about the questionnaires I've prepared for the client visit. <project manager> doesn't think it's necessary for me to question the clerk and even suggested that we cut back on the manager and mechanic interviews. I tried to offer him a couple of reasons why the interviews are important, but he doesn't understand or believe me. I asked him to trust me, but he still doesn't want me to do the interviews. | I think this event is a negative event. It shows a breakdown in communication, partly because I don't know how to express in clear terms why the interviews are important, especially when I'm on the spot. It also shows a breakdown in collaboration because <project manager> is unwilling to work with me to ensure that I can get the information I need in order to do my job well. |
| Usability Engineer | 6/23/2008 | I heard from <researcher> that <product manager>, <management 2>, and <management> had a meeting about the site visit. <management> told <researcher> that the meeting is more about getting the customer on-board and repairing the broken business relationship than gathering usability data. <management> told <researcher> that <product manager> would do most of the talking and that we probably wouldn't get to do any interviews or observation, as planned. | This was a negative incident that shows breakdown of communication and lack of collaboration. <product manager> spoke to <management>, apparently because of concern about interview content, without first contacting the rest of the team to discuss concerns and negotiate. The fact that this information came down the day before the travel just exacerbated the problem and feeling of team disunity. |
| Usability Engineer | 6/23/2008 | <developer> came by my desk and I let him know about the change in the site visit plans. He told me that <management> had been leaning on him to push ahead with developing the UI, even though the UI should be driven by usability mock ups. | This is a negative collaboration incident. <management> is aware of the development process we should be following, but is apparently trying to enforce another process without the knowledge of the team. |
| Usability Engineer | 6/26/2008 | <project manager> held a requirements meeting after our Site visit with the whole team. The meeting lasted an hour and a half, but very little was accomplished. The problem is that the meeting lacked goals and structure, as most of our team meetings. The result is that many team members were not engaged because information was not made relevant or clear to them. The teammates that were following got vague and spotty information. | The event shows a lack of ability to communicate information to the team effectively. This directly affects team collaboration in the sense that not all team members feel involved or interested in participating in the group. Well-run meetings can help to increase team cohesion, while poorly-run meetings lead to slumping involvement. I think this is part of the reason <product manager> doesn't attend our meetings. |

| | | | |
|--------------------|-----------|--|--|
| Developer | 6/26/2008 | In the sync up meeting, <qa consultant> (QA person), first told everyone that he'd be doing other work during the meeting. This would have been mildly okay, if he didn't constantly try to interrupt others sentences and tell something that's irrelevant. He also kept suggesting new features and ended up advising everyone that we should define our scope. He also kept asking questions about things that were either just discussed or was being discussed. Listening and paying attention to what's going on is important and he didn't do that. | If this kind of behavior goes on, then this will hinder our quality and we will waste time retesting or re-discussing things. |
| Developer | 6/26/2008 | After the teams <customer company> visit and my meeting with <core dev>, I was able to confirm a lot of information about <BMS> I heard from different sources. | This is really positive, now we have a much better idea of what we'll be doing. |
| Developer | 6/27/2008 | <qa manager> seemed to be willing to meet us on a more middle ground in terms which version of Meridium to use. Her concerns were fleshed out and now we know that we need to tell the customer, they shouldn't expect production software in Phase 1. | This is positive. We have a better idea of how to handle this application. |
| Usability Engineer | 7/2/2008 | <developer> just stopped by my office to tell me that he's worried<management> is leaning on <developer 2>. He thinks <developer 2>'s comments about the web application in our meeting today are<management>'s influence, and that<management> will be pushing <developer 2> to develop the interface prematurely. <developer> is interested in me getting UI mockups out as soon as possible so that there is legitimate basis for UI development. | This is a communication issue.<management>, once again, is communicating to the team without going through the team lead, <project manager>. This results in mixed messages and disunity in team goals/agendas. |
| Developer | 7/1/2008 | After having an architecture discussion with the team, PM has their own meeting and they dictate an architecture. | This negatively affects my involvement in this project. |
| | | This clearly shows that some of the requirements weren't communicated well. | |
| Developer | 7/2/2008 | PM communicates their new requirements more clearly and even though the architecture they propose has potential issues, for now it's fine and it fits the company's general vision. | Communication is always positive, but this should've been done earlier. This just makes me realize how detached our teams is from the customer, even though the team had a site visit, no one said anything about a low cost IT requirement. |

| | | | |
|--------------------|----------|--|--|
| Usability Engineer | 7/7/2008 | The team doesn't understand scenarios and I was unable to explain it to them. This came up in the meeting today when <qa manager> questioned the breakdown of what she thought was one scenario into two separate scenarios. We spent several minutes discussing this issue and I wasn't happy with the outcome. | Because of issues like this, we spend little time on meaningful issues and our meetings always run over schedule. |
| | | | |
| | | This communication problem is rooted in the collaboration problem of wasting time in stand-up meetings on minute details. One cause is that I discussed UI issues in the depth that I did; I'm not yet familiar with which bits of information are relevant to which parties. Another is that there is currently no structure to our meetings that defines topics to discuss and priorities. | |
| Usability Engineer | 7/7/2008 | <project manager> and I didn't coordinate my scenarios and his iteration plan. We were unaware that the other was doing activities that affected our own work. | This just caused confusion in the meeting as to the implications of one's work on the other's. Better communication would have allowed us to save time in the meeting and to have prevented need for rework. |
| Usability Engineer | 7/7/2008 | The team worked through our iteration plan in a team meeting today. There was a communication breakdown about when usability testing and code rework can take place in the development cycle. <project manager> initially said that all feature rework must take place within one two-week iteration and that a feature will be considered complete by the end of the iteration. This concerned me greatly because this leaves no time for usability evaluation and resulting changes. About 30 minutes later, I finally was able to hear other teammates confirm that rework can and will happen in following iterations. | This communication breakdown made me on the offensive for the whole meeting and hindered forward progress of iteration discussion. |
| Usability Engineer | 7/7/2008 | I think our iteration planning meeting went well, relative to our typical meetings. We were able to get a good stakeholder turnout and together we worked out some tangible details about our team strategy. | We were able to communicate openly, with the voices of interested parties being taken into consideration. This resulted in a plan that we can all somewhat agree on and understand. |

| | | | |
|--------------------|-----------|--|---|
| Usability Engineer | 7/7/2008 | I sent a list of high-level goals for <project manager> to forward on/present to the team last Wednesday, but he hasn't responded yet. <project manager> told me the goals were good and important, but he doesn't seem to be treating them like they are. | I'm not sure if this is a communication or collaboration breakdown, or something else entirely. But, the effect is that the team is still in great need of unifying principles and I feel like the work I put into that effort was largely useless. These breakdowns have the larger effect of making me feel as though the team is in the hands of an incompetent leader; it's hard to trust his decisions and respect his authority. For the team, the larger effect is that we don't operate like a team at all. |
| Usability Engineer | 7/9/2008 | I got an email from <project manager>: "I'm setting up a meeting to talk about just the UI. I'm thinking we can simplify this down from 4 screens." I think it shows poor collaboration to address me in an email like this; it reflects the attitude that he doesn't trust me to do my job well. I understand that UI discussions need to be had, but would appreciate the opportunity to clarify my designs before <project manager> starts making off the cuff design decisions like his suggestion to "simplify this down from 4 screens." | It hinders collaboration because I am less inclined to work well with someone whom I think disrespects my work. |
| Usability Engineer | 7/9/2008 | <project manager> disagreed with a feature in my UI mockups in the UI meeting today. Even though all other team members agreed with this feature, <project manager> continued to return back to its discussion to try to change the feature. | I think this shows negative collaboration. There was majority consensus after thoughtful deliberation, so <project manager> should have converged with the group decision. |
| Usability Engineer | 7/16/2008 | I've been getting positive feedback about my UI mockups. In the meeting, <project manager> and others said good things about the design. In my office, <developer 2> stopped by and said that the design was excellent and that he wanted to show<management>. | Whether or not the design is actually usable has yet to be determined, but the fact that my teammates are backing up my work shows good team communication (positive feedback is usually harder to come by than negative) and bodes well for future collaboration (mutual respect helps grease the wheels for quality work). |

| | | | |
|--------------------|-----------|--|--|
| Usability Engineer | 7/16/2008 | <project manager> announced at the meeting that<customer representative>wouldn't be joining us, even though I had spoken with him a half hour prior about what I would show to <customer representative>. There was a communication breakdown between <project manager> and me (and the team) as to whether<customer representative>would be present. | This is a negative event with regards to communication and, to a small extent, collaboration. The message could and should have been sent earlier to let the team know that<customer representative>probably wouldn't be at our meeting (communication). And, <project manager> disregarded the fact that I was preparing for meeting with <customer representative>, perhaps working harder than I needed to (collaboration). |
| Usability Engineer | 7/14/2008 | This past Monday, there were several impromptu meetings held between <qa> and I, <developer> and I, and <developer 2> and I. meetings were short, useful, and a couple of them were initiated by the other party (which shows that they are interested in engaging and including me in group discussion). | These meetings are positive signs of group communication and collaboration. They are targeted, including only the relevant team members, and they reify a sense of group collaboration. |
| Tester | 7/16/2008 | After our normal stand-up meeting, a few of us got together to discuss questions and concerns about the User Interface. We were able to be open about suggestions and it facilitated communication especially since it helped us focus on perfecting one aspect of the project. We remained in contact via email after the meeting to give feedback on changes and iron out the details. | It affected communication positively because we stayed in contact so we could get things looking or working better in a short period of time instead of having to wait until the next stand-up meeting. |
| Usability Engineer | 7/22/2008 | In the first iteration plan meeting (today) <project manager> addressed <product manager> with the question of how to evaluate the usability of the mockups. I had trouble getting included in this conversation. | While <product manager> should be involved in the planning and coordination of this activity, he should not be the primary person to consult about what sort of studies should take place and with whom. This is a job for the usability engineer (me!) :). This was a collaboration breakdown (include me!) and a communication breakdown (let me get my words in!). |
| Usability Engineer | 7/23/2008 | I looked at the iteration plan update () and realized that <project manager>'s placed some columns in between the Dev Time and QA/Doc/Usability Time columns. | I think this shows that Dev is more important than the rest of the team contributors. This is a knock to team collaboration because it weakens the sense of unity. |
| Tester | 7/28/2008 | Today's standup meeting was difficult to sit through because the first "final-looking" version of the touch screen project was unveiled...and it wasn't what usability had intended for it to be. There were many conflicting | I think meetings that turn out like this one did are not very productive and cause people to leave upset. From what I can tell at this point, the people involved are willing to agree as to the timeline of the |

| | | | |
|--|--|---|--|
| | | <p>opinions concerning it and at times there was a lot of contention.</p> | <p>project and various iteration checkpoint dates, but disagree as to the scope of the project. There was contention over whether the coding of the program should be "<customer company>-specific" or general (to facilitate applying this failure code touch screen to other companies with different needs). I thought that we were going to make it general and had agreed on that but from the meeting it appears that the issue is not closed. There apparently isn't agreement between usability and development as to how the user interface is supposed to work and I think it's frustrating that the vision for how the UI works isn't clear at this point. I thought, just as usability did, that we were going to avoid the keyboard/mouse interfaces as much as possible with the exception of comments (keeping 95% of the interaction between the user and the interface, touch screen-related). The meeting would suggest that perhaps the user will use the keyboard 50% of the time...which lessens the importance of making this a touch screen interface. It would be nice if we all had the same vision for where this project is going and spent more time having individuals meet to iron out details, instead of having a bunch of meetings where we're guessing what would be best for <customer company> while we spend multiple meetings discussing the calendar and our time plan for completing the project. There is communication among the members of the team but I don't think that it is very effective...since the communication is superficial in many of the meetings and then after part of the program is already complete, everyone voices their opinions about why they don't like it. The team needs to be unified in their vision as to what we need to accomplish and how the project should run...and right now it doesn't look like we are.</p> |
|--|--|---|--|

| | | | |
|--------------------|-----------|---|---|
| Usability Engineer | 7/28/2008 | In the stand-up today, <developer> presented the developed interface. In one of his features, he did not follow the UI mockups. The feature I am referring to is the Operation selection text display. From my perspective, it would not have taken any extra time to implement the feature the way it is shown in the mockups. | Because <developer> gave a usability counter argument in defense of the way he developed the UI, it seems to me that he deliberately disregarded the mocukups for this feature. I think this is a lack of collaboration. |
| Usability Engineer | 7/28/2008 | In the stand up meeting today, a group discussion over a UI feature took place. I got frustrated that team members were making design claims and suggestions, mostly because I didn't think that it was the right time/place to be discussing them and that the discussion was taking place over a feature that wasn't designed to my specifications. I snapped at <qa manager> when she decided to offer up one last suggestion. | I think this is a sign of both miscommunication and poor collaboration. This feature had already been discussed in meetings that not all members attended; apparently this information is important to those non-attending team members (or is this a managerial problem, where the bounds of responsibility are not well-defined?). Also, collaboration broke down when I cut off <qa manager> from giving a suggestion. |
| Usability Engineer | 7/30/2008 | <documenter>, <qa>, and I had an impromptu meeting in the kitchen for about 45 minutes after our stand up. We discussed the usability study, the new system design, possible enhancements, etc. This session allowed me to bounce ideas off of them and get new ideas for the interface. | Built sense of collaboration and team cohesion; helped me do my job better |
| | | | |
| | | <qa person> had gone through and calculated the difference in the number of "clicks" for each screen from the old mockups to the new. He took into consideration the clicks needed for a successful interaction vs. a mistake/correction interaction (maybe he should have my job!?). This was really helpful for realizing the potential to speed up the interaction and provides some tangible rationale for making UI changes. | |

| | | | |
|--------------------|-----------|---|---|
| Usability Engineer | 7/30/2008 | <developer> approached me after the meeting today to elaborate on why he doesn't want bug fixes created for features that are not finished being implemented. My feeling is that he wants me to write bug fixes for Iteration 1 items and go against the instructions given by <project manager> in the meeting. <developer> gave me some rationale for his opinion about how this work should be done that wasn't brought up in the meeting. When I told him to explain this rationale with <project manager>, <developer> looked dismayed and our conversation petered out. | There is a break in communication between <developer> and <project manager> on this issue. The matter was not resolved in the meeting, although it seemed to have been. <developer> also seems to be trying to push his solution without going through <project manager> for approval. |
| Usability Engineer | 7/30/2008 | I spoke with <developer> this afternoon about some of the impending UI changes. He is frustrated that the mockups keep changing. He says that to keep programming this way, he'd have to work 70 hour weeks and it's not worth it. I brought up the point that <project manager> said "rework" like these mockup changes is accounted for in the Iteration Plan. <developer> said that it really isn't accounted for and that <project manager> knows it. <developer> seems very frustrated. | This incident shows that there is a misunderstanding between <project manager> and <developer>. Or, if <project manager> already knows that rework isn't included in the time estimates, then there is a miscommunication between <project manager> and the team (or perhaps just me). This is making our overworked developer frustrated and it makes it hard for me to judge when I have to deliver work. |
| | | | |
| | | Side note: I've sent <project manager> an email to set up a meeting to discuss when I should deliver stable mockups and how this fits in with rework in the Iteration plan. | |
| Product Manager | 8/1/2008 | Some team members were trying to make requirements/Functionality/usability decisions without data to back those decisions up. | Caused confusion on the real requirements and can lead to developing the wrong functionality. |
| Product Manager | 8/1/2008 | The developer is not being kept up to date with changes to the UI. Therefore when he presents his work he is criticized for something he knew nothing about. | Leads to poor team interaction and fosters animosity. Involving the developer in these changes would help with this situation. Agile is supposed to involve all team members. |
| Product Manager | 8/4/2008 | There is a lack of understanding on how to write tickets and how tickets should be coded in the WT system. | This makes it hard for everyone except the creator of the ticket to know how to handle the tickets and what the priority/severity of the tickets are. |
| | | | It also wastes time in the team meetings. |

| | | | |
|--------------------|-----------|---|--|
| Product Manager | 8/4/2008 | Discussion and disagreement on whether a ticket should be written as a bug or an enhancement. Conflicting view and changes in the process. A decision was made, reversed then changed back. | Team does not have a clear understanding of how to manage tickets or how to write tickets with all of these changes. A clear and consistent understanding at the start of a project would solve this. |
| Developer | 8/18/2008 | Standup meeting is late. It's 9:39 and <project manager>'s not here. It's just me, <product manager>, <qa consultant>. | Cuts in on team awareness |
| Usability Engineer | 8/18/2008 | I just realized that we haven't received some information from<customer representative>that we requested a month or so ago. We asked to have several example time sheets filled out. I don't think this is the first time that we've not followed up to ensure we get what we need. | This is more of a management issue than a communication or collaboration issue (unless <project manager> is deliberately forgetting to follow up on this). I think it's important, though, because it shows that some of the efforts we've taken (to elicit this need and communicate it to the customer) have gone to waste and no one seems to have noticed. |
| Usability Engineer | 8/18/2008 | <developer> called me over to his desk and walked through the prototype he developed while I was gone. He insisted on going through it with me so that we could talk some things through. As he showed me the functionality with a walk-through, we exchanged information about feature pros and cons and occasionally alluding to some study results to back up feature claims. I was surprised at how much he conceded to my opinion and agreed to make changes. He took notes about the feature changes I requested and rolled those into his list of change requests that he presented at the feature team meeting. | This sort of communication is much more effective than simply creating work items and assigning them to <developer>; it helps us reach fair consensus and I project that it helps <developer> better remember which features need to be changed (and perhaps allows him to strategically fix them at a higher rate). |

| | | | |
|--------------------|-----------|---|--|
| Usability Engineer | 8/20/2008 | <p><project manager> set up a meeting for Phase II UI brainstorming. He's invited <product manager>, <core dev>, <developer>, himself, and myself. The point of the meeting, according to <project manager>, is to help define a general template for the interface that can be used once we "move all of Meridium to <web technology>." My concern is that this is going to be a "design by committee" sort of thing. I'm also worried that design will suffer for the consideration of a future Meridium <web technology> interface that will never come to fruition (since it seems unlikely that we'll be moving to <web technology> in the near future). Furthermore, I don't think that these matters can be constructively addressed in an hour long session with 5 people in attendance. I communicated these ideas to <project manager>, who basically said "I understand, but I think it will be useful".</p> | <p>I feel this situation is tied to a miscommunication / misunderstanding / lack of collaboration with respect to the balance of job responsibility between <project manager> and myself. From my perspective, "UI brainstorming" meetings should not be held; this particular meeting should be more focused on sharing concerns and needs between parties so I can incorporate them into design.</p> |
| Product Manager | 8/25/2008 | <p>At this time the customer (<customer representative>) is not involved due to vacation. The team is starting to make decisions on the interface and functionality without any data to back up the decision. In the conversations you hear a lot of "I think...", "Why don't we..."</p> | <p><researcher> I have written this same type of CI up before. It seems to be a recurring theme. You must have data to base your decisions on. Usability decisions should be based on user studies. Functionality decisions should be based on requirements.</p> |
| Usability Engineer | 9/8/2008 | <p><developer> developed a feature on the interface differently than was prescribed in my mockups. When we were discussing this feature last week, he told me his idea for how to implement it; I then told him my idea and why it was preferred and then sent him a mockup a few minutes later. <developer> deliberately ignored my design and coded it his way.</p> | <p>This is an issue that has come up multiple times. <developer> thinks he knows better than I how the UI should behave, which insults me professionally. It shows a lack of cooperation, a lack of trust in my ability, a lack of communication on his part. I think this hurts team unity as well as team individuality (the ability for me to feel like my work is meaningful and appreciated), and thus hurts team morale; I feel less inclined to produce mockups because I see how they are disregarded, and I also feel less inclined to care when the project is going in the wrong direction.</p> |
| Product Manager | 9/10/2008 | <p>lack of documentation during meetings with the customer. I have been making some notes</p> | <p>You do not have insight into why some decisions are made and you also end up going back and re-debating some decisions.</p> |

| | | | |
|-----------------|-----------|---|--|
| Product Manager | 9/19/2008 | Attitude. Attitude cannot be under estimated. It has a direct effect to the performance of the team. When a bad attitude surfaces the team shutdown. In this case and at other times in this project the team lead has come into meetings with a bad attitude which has completely disrupted the meeting | Shuts the team down. |
| Product Manager | 9/18/2008 | Meeting without a clear direction or purpose. | waste of time. |
| Product Manager | 9/18/2008 | The review of the phase one items was pointless. This was done on the code complete date so there was no room to add any work to phase one. This should have been done earlier in the process to allow for work to be done in phase one. | Does not allow the team to appropriately address work. Between feature complete and up to code complete tickets need to be reviewed on a regular basis not once at a meeting on code complete. |

Observed Critical Incidents from Touchscreen Project.

These incidents were observed, documented, and sorted by the researcher.

External effect incidents

| Role | Date of incident: | Event description | Effect of incident |
|--------------|-------------------|---|---|
| | | External company requirements | |
| Project lead | 7/2/2008 | At the daily meeting, project manager revealed that the system will be a web interface. He just received this requirement from his boss. Web interface is strategically important for Meridium as it will have low install/maintenance costs. | New and changing requirements such as this will have implications for development. This requires the developers to learn new technologies and may put pressure on the release date. Agile processes such as the one followed here are supposed to accommodate such changes. |

| | | | |
|---|--------------------------|--|---|
| Usability Engineer, QA manager, Project manager. | 9/5/2008 | At daily meeting, QA manager says need some way to identify version of the product so if doing customer service, can tell customer to look at version #. Proj. Mgr. asks UE to work on a way to do that. | External requirements for products (not necessarily requested by customer) - requested by other depts. May require changes to UI. |
| | | | |
| | | External dependencies on other systems | |
| Role | Date of incident: | Event description | Effect of incident |
| Project Manager, Developer | 9/13/2008 | At daily meet, Proj. Mr. says supposed to be code complete this weekend but not going to happen. Asks for status from Developer. Developer says there are bugs in core connector and asked core team to fix it. Their fix broke a lot of our team's code. | External dependencies can hinder project development. |
| Project lead, developer | 6/23/2008 | Developer asked company <BMS> expert about how to use the <BMS> connector but did not get a timely response from him. Wants a technical contact from the customer side so he can get this type of information. Project lead tells him to just wait for a response from our guy before looking for a technical contact from the customer. | Relying on expertise/skills from people outside of the team may hinder project progress. |
| Developer | 7/2/2008 | At daily meeting, lead developer spoke with person in charge of <BMS> connector and found that it was buggy and may have issues. Developer hasn't had chance to use connector to get data. | Major blocking issue not related to usability but which may impact what can be implemented in terms of user experience. |
| project lead, product lead, Developer, UE, QA manager | 7/14/2008 | At daily meeting, Project lead asked Product manager to get in touch w/ <customer company>--> with their IT department. Need to know whether their IT department will support <tech>. This is the only major blocking issue with using <web technology>. Until we know for sure, development using <web technology> cannot proceed. | Blocking issues not related to development/usability within the team can hinder project velocity. |
| Usability engineer, Project manager | 9/8/2008 | UE says would be best to have customer data to run test on so can see how UI looks with real information. Project manager agrees. Says <product manager> sent email to customer to get export of data. | External dependencies can hinder project development. |

| Role | Date of incident: | Event description | Effect of incident |
|----------------------------|--------------------------|---|---|
| | | | |
| | | Other external effects | |
| Project manager | 8/6/2008 | Project manager gets email from customer saying the touchscreen he was sending somehow got lost in the shipping process. Now team does not have access to touch screen. May need to get hardware elsewhere. | Not having an actual touchscreen may have acceptance testing and usability testing implications. Could hinder development progress. |
| Usability engineer | 8/8/2008 | Usability engineer involved in car accident so did not get into office. | Technology failures may hinder development progress. Can hinder communication & collaboration with team. |
| Project Manager | 7/16/2008 | At daily meeting, found out that customer would not call into meeting today. Project manager saw that customer was at home that day caring for sick spouse. | Events outside control of team members may hinder project communication/collaboration. |
| | | | |
| | | Company culture affecting development | |
| Role | Date of incident: | Event description | Effect of incident |
| Product manager, usability | 5/29/2008 | need to get customer buy-in as soon as possible so we can have someone from company to be involved in team to verify requirements, approve changes to requirements, do acceptance testing. Phone meeting with customer--customer is excited to be involved but cannot get him involved immediately b/c he does not have authority to approve process. Need to get through bureaucracy to get customer buy-in. (<product manager> needs to get to <management 2> who needs to go through<customer representative>at <customer company> who needs to go through his bosses) | Having no customer involvement will limit what can be done in terms of agile software development and usability. |

| | | | |
|--|-----------|--|---|
| Product manager | 6/9/2008 | At the iteration 0 meeting, Discussed progress in getting customer involved in the project to the level we requested. <product manager> mentioned that he needs to get in touch with <management 2> who is in contact with people in <customer company>. There is reluctance to get customer involved before we have something tangible to show them wrt requirements/prototype/signed contract. | Bureaucracy within the company seems to hinder the type and level of user involvement that is needed for agile usability processes. With limited customer involvement, we may not develop system that meets their specific needs. |
| Product manager, usability engr | 6/19/2008 | Usability engineer needs to get salary information to do calculations to estimate total savings to customer. Project lead and product manager says it is bad form to ask customer about salary information. This will make it difficult for usability engineer to measure project progress and project success. | This again goes back to product manager's hesitance to communicate openly with customer and work with customer to develop product that meets their needs. May need to rely on customer to get metrics (e.g. must complete in 5 minutes or less) |
| VP of development , Project lead, myself | 6/23/2008 | Meeting with VP of development to discuss site visit. He is concerned because Meridium has not sold client on the project. This is more of a meet and greet and does not want to scare away customer by throwing a big list of usability/interview questions at them. Want usability person to lead product lead take the lead and have the usability person largely stay quiet. | Limiting the work of the usability engineer--limiting her ability to collect data necessary for project success threatens the success of the project. Threatens the ability to deliver a usable project. |
| Usability engineer, product lead | 6/23/2008 | At meeting with VP of development, we find out that product lead has already cut out some of the interview questions prepared by the usability engineer. He is concerned about being sensitive to the customer and not scaring him away. This was done without consulting the usability person or anyone else on the team. | Not sharing information and being open hinders collaboration and threatens relationships/trust between team members. |
| Usability Engineer, Project manager | 7/25/2008 | Project manager does not want UE to do evaluations with real end users b/c it may 'step on toes'. Wants to be sensitive to customer concerns (people paying for the system are different from the end users who will be using it). | Development culture company is one where customer should be shielded from product as it is developed. Only want to show finished/polished product to customers so we seem as professional as possible. This culture limits/hinders usability effectiveness and agile development b/c of importance of close relationship with customer. |

| | | | |
|-------------------------------------|-----------|---|--|
| Product manager, Usability engineer | 7/28/2008 | At UI meeting with Product manager, UE mentioned that developer is concerned about showing interactive mockup to <customer representative>, and then showing less interactive/pretty version later when it's actually implemented. Product manager says just to show non-interactive version to customer. | Company culture seems to be to 'shield' customer from incomplete product to maintain air of 'professionalism'. Compromise by UE to show mockup to customer in a way that is not confusing. |
|-------------------------------------|-----------|---|--|

Technical issue incidents

| Technology failure hindering communication (meeting) | | | |
|---|--------------------------|--|--|
| Role | Date of incident: | Event description | Effect of incident |
| Project manager | 6/27/2008 | Project manager arrives a little late and has trouble getting his laptop screen to show up on the projector. | Technical difficulties can hinder project progress and may be difficult to overcome if certain team members of if the customer is participating remotely |
| Developer | 7/11/2008 | At the daily meeting, developer showed a basic implementation he did of UE mockup using <web technology>. It did not look correct because the resolution of the projector was different from the resolution he used to develop it. | Technical difficulties can hinder project progress and may be difficult to overcome if certain team members of if the customer is participating remotely |
| Project lead | 7/14/2008 | At daily meet, Project lead wants to review iteration plan with team but project bulb burned out and he could not. Ended up just ending meeting early. | Technology failures can hinder communication during meetings. |
| Developer | 8/6/2008 | At meeting with customer, Developer demoed current implementation. Developer was running it remotely from his machine and it was trying to run from a virtual machine on his desktop--result was very slow performance in the demo. Developer explained to customer why performance was so slow. Customer still seemed happy to see actual functioning system. | Technology-related problems related to communication tools used (remote desktop, screen sharing software) can misrepresent state of the implementation and confuse the customer. |
| Project manager | 8/11/2008 | At daily meeting, Project manager wants to review work tickets through team system. Has problem with query to display all currently running work items. Spends several minutes trying to get it to display correctly. | Problems with ease of use of tools could waste time in regular daily meetings. |

| | | | |
|--|-----------|--|--|
| Project manager | 8/20/2008 | At customer demo, had problems getting everyone on the <screenshare software> screen sharing program to do the demo. There are several different people from customer company joining meeting to evaluate product for further adoption. This seems to be a technical issue with the screen sharing program. | Technical difficulties can hinder project progress and may be difficult to overcome if certain team members of if the customer is participating remotely |
| Developer, Project manager | 8/25/2008 | At daily meeting, developer tried to demo picklist stuff to rest of the team. Did not run on project manager's computer. | Technical issue hindering communication of development progress to rest of the team. |
| Project manager, developer | 8/25/2008 | At feature team meeting, Project manager wanted to review the live system but it still did not work correctly on the computer he tried to access it on. Developer says screen resolution might be wrong and should full screen the application. After making the application go full screen, it worked correctly. | Technical issues related to support technologies may hinder development progress and/or communication. |
| Project manager, Developer | 9/3/2008 | At iteration planning meeting, customers are at 3 different locations (two others including one <BMS> guy). Developer tries to use new screen sharing program b/c of limitation of <screenshare software> (handles only three total). Takes ten minutes to try to set it up but it fails. Goes back to <screenshare software> w/ one customer connecting through that & other 2 customers using netmeeting to connect to first customer. | Technical difficulties can hinder project progress and may be difficult to overcome if certain team members of if the customer is participating remotely |
| Developer, QA Manager, Project manager | 9/5/2008 | At daily meeting, developer and QA manager spend several minutes fixing the work items query on VS. It somehow got messed up. Proj. manager wanted to review high priority work items for developer. | Technical difficulties can hinder project progress and may be difficult to overcome if certain team members of if the customer is participating remotely |
| Project Manager | 9/8/2008 | Project manager comes into feature team meeting late. Comes to roanoke training room saying the mail program sucks. Says he reserved <meeting room> but didn't actually get reserved. | Technical issues can hinder project progress. |
| | | | |
| | | Technical issue affecting project progress/velocity | |

| Role | Date of incident: | Event description | Effect of incident |
|--------------------|--------------------------|---|--|
| Usability engineer | 6/15/2008 | Usability engineer attempted to access the sharepoint Portal through the VPN connection but was unable to do so. She was therefore unable to easily access project documents. | Being unable to access project documents will make it difficult to be productive when away from the office. This is a technology limitation (UE person works on a Mac laptop). |
| Developer | 7/21/2008 | Developer previously tried to edit the iteration plan document on the portal but the changes wouldn't save. Eventually just created another copy of the document and uploaded it. | Technology failures can hinder communication during meetings. |
| Usability engineer | 8/6/2008 | Laptop notebook that usability engineer has been using failed so didn't go into office. Had to get it fixed because she had a conference to go to next week and needs laptop for that. | Technology failures may hinder development progress. Can hinder communication & collaboration with team. |
| Developer | 8/20/2008 | At customer demo practice meeting, developer says other developer resource was tasked with doing some backend work but couldn't get it done b/c of technical issue (database differences). Since other developer is in <loc> and he has poor internet connectivity, won't be able to do work until later. | Technical issues would limit productivity of developers. Problem is exacerbated by fact that he is not colocated with rest of team. |
| Developer | 8/25/2008 | At daily meeting, developer got picklist work item done, but spent most of last Friday upgrading Meridium build--encountered issues in upgrade. | Technical issues related to support technologies may hinder development progress. |

Communication Incidents

| Role | Date of incident: | Event description | Effect of incident |
|-------------|--------------------------|---------------------------------|---------------------------|
| | | Generating common ground | |

| | | | |
|--|-----------|--|---|
| Project manager, QA manager | 6/11/2008 | At standup meeting, <project manager> mentioned that he wanted each 'iteration' to result in a piece of functionality that has been designed, implemented and tested. <qa manager> says that is not practical as code is checked in at the end of each 'iteration'. <qa manager> and <project manager> have different understanding of what an 'iteration' is. She thinks an iteration (which is 1 week) results in a RC being delivered that can then be tested. <project manager> is saying an 'iteration' is two weeks and includes both. | Lack of common ground will result in miscommunications and misunderstandings that could also hamper team collaboration. These issues need to be identified and resolved so all team members are using the same terms in the same way. |
| Project manager, QA manager, Documentation | 6/13/2008 | At the daily meeting, the project manager went over what he thought was a misunderstanding over what was meant by the term 'iteration'. There was a perceived difference between what he meant and what QA meant by 'iteration'. There was general agreement that they just need to be sure that everyone means the same things by the terms they use in the project. | Lack of common ground can result in confusion and hamper collaboration--especially if they aren't uncovered and addressed in a timely manner like was the case this time. |
| Usability engineer, product lead, project lead | 6/25/2008 | At site visit, team members were able to generate common ground with customer. Learning customer-specific terminology/lingo. | Collaborating with customer as active team participant means it is important to generate common ground to aid in communication. Site visit helped to do this. |
| Project lead, usability engineer | 7/2/2008 | At the daily meeting, usability expert presented draft of the workflow to give other team members understanding of how things work and how things will change w/ introduction of touchscreen. Proj lead asks where 'work order' is in the process. Usab. Says she is calling it the 'error codes/history form. | This demonstrates a lack of common ground between team members. This difficulties can be overcome with regular meetings and close team collaboration. |
| Usability, QA manager | 7/7/2008 | At the daily meeting, there is disagreement about what is meant by the term 'scenario'. QA manager does not think the initial list of scenarios that the usability expert came up with distinct 'taskflows'. There is a discussion about what is meant by 'scenario' and what should be in the list of scenarios. | Shows a lack of common ground between team members. This difficulty can be overcome with regular meetings and close team collaboration. |

| | | | |
|---|-----------|---|---|
| Project lead, QA manager | 7/7/2008 | at iteration planning meeting, In trying to plan what goes in each iteration, project lead and QA manager decide to use new terminology that is different than terminology previously used at company. This is to make the terms for descriptive and not confuse people on the project and outside the project. ('feature complete'-->iteration feature complete; 'code complete'-->iteration rework complete). | Lack of common ground will result in miscommunications and misunderstandings that could also hamper team collaboration. These issues need to be identified and resolved so all team members are using the same terms in the same way. |
| Project Manager, QA | 7/18/2008 | Project manager shows others (UE, doc, QA) how work items work in VS. QA asks what a 'change request' is. Project manager says to think of it as a 'feature' and to ignore the current name. | Common understanding of terms/definitions among team members is important for effective communication within the team. |
| Project manager, Qa manager | 7/22/2008 | At Iteration 1 Review meeting, project manager mentions that what QA manager is calling 'workflows', the rest of the team is referring to as 'scenarios' | Lack of common ground will result in miscommunications and misunderstandings that could also hamper team collaboration. These issues need to be identified and resolved so all team members are using the same terms in the same way. |
| Project manager, Product manager, usability engineer, QA manager | 7/22/2008 | At Iteration 1 Review meeting,QA manager does not want to show mockup to customer too soon, b/c/ it is not stable and will change as development proceeds. Product manager agrees-does not want to show customer 'unpolished' UI. Project mgr and UE agree. Showing this type of thing early to customer will help get feedback early on and will show that we're working with them to refine product. | Some members of team do not understand some UE/agile methods (close collab. With customer and early evaluations). Existing culture is to shield customer from product as it is developed and only show 'polished' product at the end. This disconnect could cause problems with project progress and how well we collaborate with customer. |
| | | | |
| | | | |
| | | Communications related to project roles | |

| Role | Date of incident: | Event description | Effect of incident |
|--|--------------------------|--|---|
| Project manager, QA manager, Documentation | 6/11/2008 | At the daily meeting, project manager again went over the definition of what an agile project was and what the high level goals were for the NSF project. QA manager stopped him and mentioned that she felt like she was being lectured to. Project manager said he felt he did not have buy in from QA and doc. QA and doc stated that they did have buy in. | There was a misunderstanding between project manager and the rest of the team as to whether there was buy in about the agile process that was being followed. This was resolved in the meeting. |
| Usability engineer, Project manager | 6/16/2008 | At the daily meeting, the usability engineer mentioned her difficulty in accessing the portal through the VPM. Project manager tells her to talk to IT guys to get access. | Open communication among team members will improve efficiency and promote more effective collaboration. |
| Developer, QA manager | 6/20/2008 | Developer started exploring how to use backend <BMS> connector working (code spike). Suggests that the project use <newer> connector code b/c it fixes a security problem with <older>. QA manager mentions that they can't use <newer> b/c it will be released after the touchscreen phase 1 is released to customer. Says it will be bad idea to use RC code in phase 1 as it may change before <newer> is released and could cause upgrade/install problems. Suggests we go with <older> for now. | Daily meeting used to discuss problem associated with underlying technology that will be used. Good team communication and discussion of issue that needs to be addressed by the project. |
| Project lead, usability engineer | 6/23/2008 | Usab. Eng. And project lead meet to discuss questionnaires she prepared for site visit. Project lead asks if usability eng. Can just plan on talking with primary customer contact, instead of interviewing all the people involved in the workflow. Says may not have time to interview everyone. Usab. eng. states that it is important to interview everyone involved. They come to understanding that info. gathering is to get actual data rather than rely on assumptions. Usab. eng. will collect what she can. | There is disconnect between what project lead thinks is needed to design the UI and what the usability engineer thinks is needed. Ended up coming to compromise in the meeting. |
| Project lead, usability engineer | 6/23/2008 | Usab. Eng. And project lead meet to discuss questionnaires she prepared for site visit. Project Lead notices that she still hasn't gotten wireless access through the company VPN and asks her to do it again. | Effective team collaboration relies on each team member working with every other team member effectively. |

| | | | |
|---|-----------|--|--|
| Developer, QA manager | 6/27/2008 | Developer again suggests that the project use <newer> connector code b/c it fixes a security problem with older>. QA manager mentions that they can't use <newer> b/c it will be released after the touchscreen phase 1 is released to customer. Says it will be bad idea to use RC code in phase 1 as it may change before <newer> is released and could cause upgrade/install problems. Suggests we go with <older> for now. QA does not want customer doing actual work with a product that has not officially been released. need to communicate to customer that initial release is just a 'beta' and actual work should not be done on it. | Daily meeting used to discuss problem associated with underlying technology that will be used. Good team communication and discussion of issue that needs to be addressed by the project. |
| Developer, Project lead, Product lead | 6/27/2008 | Meeting with parts of team and with resident database expert to address issue of how to release 'beta' to customer within the <newer> release cycle. This addresses QA manager's concerns with release. Consensus is that these concerns need to be brought to the customer and their expectations need to be aligned with ours. | Regular meetings are effective to identify and find ways to resolve issues as they come up early on. |
| Usability | 6/30/2008 | usability person discussed UI goals separate from other goals. Thinks of them as separate concern. | Project goals should be unified and prioritized so the team can make informed decisions about where to put effort. This should include usability/end user goals and Meridium goals. |
| Project lead, usability engineer | 7/9/2008 | At UI demo meeting, project lead wants to limit # of pages and have all information on a single page. This is his personal preference. UE divided data entry into separate pages so pages aren't crowded. ->After some discussion, went with UE design. | Different team members have different idea of what is 'simple'. Team lead is not a representative user of the system and should not affect system design with personal preferences. UE person is most qualified to make design decisions. There is risk of 'design by committee' |
| Developer, usability engineer | 7/9/2008 | At UI demo meeting, Developer mentions that 'moving of back, next' buttons from screen to screen may be confusing because the buttons are not in a consistent location from page to page. This is an additional possible downside. UE wants to try with original design first and then if there is a problem try the developer's idea. Developer also mentions technical limitations with moving buttons dynamically. | UE person is most qualified to make design decisions. There is risk of 'design by committee' although in this case the developer identified a real potential problem. |

| | | | |
|--|-----------|--|--|
| Project lead, developer, usability, QA manager | 7/9/2008 | At UI demo meeting, project lead mentioned he wants to add 'prev' and 'next' button and keep them in same place on each page for simplicity & efficiency. There is concern that this will make it less obvious and easy to use. Project lead says requirement is for 'minimal trailing', not 'no training'. | Although passing reference is made to one of the goals, the project goals and priorities were not used in the discussion. Also, there is a risk of 'design by committee'. |
| QA, Usability Engineer | 7/14/2008 | At daily meet, QA started coming up with test plan based on scenarios developed by UE. Had questions on field validation and what happens in UI in certain error conditions. Met with UE after standup to discuss those issues. | Process works so that different team members can work in parallel (QA, Dev, UE). Close collaboration/communication ensures conflict areas are resolved quickly. |
| Developer, usability engineer | 7/16/2008 | At daily meeting, while UE is demoing latest version of iteration 1 prototypes, developer mentions that Work Center # is not actually a number but is free text--so shouldn't do calculator popup for that one. | Developer uses knowledge of backend <BMS> database to identify error in mockups. Useful to support collaboration. |
| Usability engineer, Documentation | 7/16/2008 | At UE/Doc/QA meeting, Doc. Mentions that having page titles will make documentation easier b/c will be easier to identify specific pages. Also mentions that it may be easier from a usability standpoint if some thumbnails in the navbar look visually similar. UE is hesitant at first but later makes this and some other suggested changes through email communication after reviewing tradeoffs. | Different team members have different goals/motivations which may affect direction of design at times. Close collaboration and careful consideration of tradeoffs to ensure that high level goals are met for end user (both from usability standpoint and from doc. standpoint) |
| Usability engineer, Documentation, QA | 7/16/2008 | Qa and Documentation schedule a separate meeting with UE to review mockups and give comments/suggestions. UE records comments primarily as tradeoffs (claims) which will be evaluated based on priorities of issues identified by Qa and Doc | Issues and concerns from other team members can be captured in claim tradeoffs and evaluated later based on priority of the problems. Helps lead to evidence-based improvement of UI (rather than design by committee) |
| Project lead, developer | 7/16/2008 | At the daily meeting, after UE presents demo, project lead asks developer about feasibility of implementing the mockups. Developer says no problem if we use <web technology>. Otherwise, it will be a big problem to do. | Need for close collaboration between UE and developer in implementing the system (usability within implementation constraints). Limited by whether or not developer will be allowed to use <web technology>. |
| Project manager, Usability engineer | 7/25/2008 | Project manager indicates that some members of the team do not understand purpose of some of the tasks the UE is doing. They appear to be a waste of time. (e.g. claims) | Poor understanding of work of different members of the team will hinder effective collaboration and may increase resentment/misunderstandings among team members. Poor communication will prevent these misunderstandings from being resolved. |

| | | | |
|--|-----------|---|--|
| Usability engineer, developer | 8/18/2008 | At daily meeting, Developer wanted to personally demo the current implementation to the usability engineer. (work that was done the 1.5 weeks she was away. Usability engineer was able to voice concerns about interface that differed from her mockups. | Impromptu meetings between UE and SE are important to clear up misunderstandings and keep UE and SE work in sync. Prevents too many disconnects. |
| Developer, usability engineer | 8/25/2008 | At daily meeting, there was a disagreement between developer & UI abt importance of certain proposed changes. Developer seems to think UI changes are just 'cosmetic' & wants to fix critical system issues/bugs first. | Poor understanding of work of different members of the team will hinder effective collaboration and may increase resentment/misunderstandings among team members. Poor communication will prevent these misunderstandings from being resolved. |
| Project manager, UE | 8/25/2008 | At daily meeting, Project manager asks UE if it's possible to have scroll bar in the picklist popup. UE gives rationale for why it's not there. Says it will be tested in next evaluation. | Issues and concerns from other team members can be captured in claim tradeoffs and evaluated later based on priority of the problems. Helps lead to evidence-based improvement of UI (rather than design by committee) |
| Project manager, usability engineer, developer | 8/25/2008 | At feature team meeting, Project manager has opinion of how 'save' button on a screen should behave. Developer & UI person have opposite opinion for how it should behave (based on technical and usability reasons) UI says let's do it how we think it should behave and then test it later. Project manager says he can't argue with that. | Validation of features through usability testing with end users is one of best ways to make design decisions with respect to usability. |
| test manager, usability engineer | 8/29/2008 | At daily meeting, test mgr mentions that one of the buttons looks disabled and doesn't look submitted. UI person agrees and says she'll look into changing it since in the usability evaluations people were somewhat confused by it. | Feedback from other team members can be incorporated into design if first reviewed by UI person. |
| Project manager, developer, usability engineer | 8/29/2008 | At daily meeting, Project mgr. says he wants summary table on a page to take the width of the screen. Developer says it will be convenient to user to keep functionality as it is. UE says she'll make the change but it should be validated in user testing. | Different team members have different idea of what is 'simple'. Team lead is not a representative user of the system and should not affect system design with personal preferences. UE person is most qualified to make design decisions. There is risk of 'design by committee' |

| | | | |
|--|-----------|---|--|
| project manger, usability engineer | 8/29/2008 | At daily meeting, project manager mentions there's a problem w/ automatically popping up an entry screen when a user enters a page. Suggests a solution he personally would prefer. UE says she will think of a solution and email it later. | Different team members have different idea of what is 'simple'. Team lead is not a representative user of the system and should not affect system design with personal preferences. UE person is most qualified to make design decisions. There is risk of 'design by committee' |
| Usability Engineer, Developer | 9/7/2008 | Last Friday, UE tells developer she's planning to run a study on Monday so she asks what would be done by then. Developer says the keyboard would be implemented and would be stable by then. When planning the study over the weekend, the build was not working. Called developer and developer told her how to get it working on her machine. | Potential issues resolved through open communication channels. |
| Project manager, developer, product manager, usability | 9/13/2008 | At daily meeting, project manager asks developer how long it would take to fix the height on lists (to make them easier to hit with the touchscreen). Developer says 1.5 hrs. Proj. Mgr. asks if he can just increase font size to make heights bigger. Developer says yes but Product Mgr. says then you'd have fewer characters shown on screen. Says he would rather have Developer do it right. Usability engr. agrees. | Each team member should focus on his own area of expertise (generalizing specialists) but still remain open to suggestions from other team members. |
| Product manager | 9/19/2008 | Product manager questions use of 'leave blank' selection on popups. Doesn't make sense in some cases. UI person not there to comment. | Absence of usability person may result in 'design by committee' situation where resulting UI is inconsistent with overall vision for system. |
| Documentation | 9/19/2008 | Documenter says selected button looks grayed out instead of selected. UI person is not present to comment on design rationale. Project manager says wait for users & see if they comment on it. | Absence of usability person may result in 'design by committee' situation where resulting UI is inconsistent with overall vision for system. |
| Developer | 9/19/2008 | Developer shows scroll functionality for components page. Functionality is similar to but inconsistent w/ other scroll functionality in the application. UI person not there to comment. | Absence of usability person may result in 'design by committee' situation where resulting UI is inconsistent with overall vision for system. |
| | | | |
| | | | |
| | | Communications with customers to resolve unknowns | |

| Role | Date of incident: | Event description | Effect of incident |
|--|--------------------------|--|---|
| Usability engineer, product lead, project lead | 6/25/2008 | At site visit, learned many details about the actual current work process, how the work materials that were previously emailed are actually used. Also learned about high level goals, project constraints and cleared up assumptions. | Demonstrates importance of communication with customers/end users to get information about what they need and want and shows how important it will be to keep them involved throughout the project--since not all questions/issues were resolved at the site visit. |
| Usability engineer, product lead, project lead | 6/25/2008 | Usability engineer previously prepared questionnaires for different end users/stakeholders based on current understanding of users/processes. Opportunistically filled out questionnaires during site visit--working with product lead and project lead to do so. | Planning and moving forward based on current, limited understanding of users will be important to work in an agile project. (bounded rationality) |
| Project lead, usability | 7/9/2008 | At iteration planning meeting, Project lead asks customer whether they want option to just add time or just enter history. Customer says yes. Usability engineer then asks why that is the case. | Usability engineer wants to know what the customer wants and also why he needs/wants certain things. Important for meeting his needs in the design of the system. |
| UE | 7/9/2008 | At the meeting with the customer, demoed the current mockups developed by UE. Cleared up misunderstandings that team had with the workflow (what should be included/added/changed) | Close collaboration with customer can catch potential issues/problems early on and save effort and lead to continuous improvement to product that will end up meeting customer needs. |
| UE | 7/9/2008 | At the meeting with the customer, made changes/edits to labels on each screen. Not major problems with current planned workflow. | Close collaboration with customer can catch potential issues/problems early on and save effort and lead to continuous improvement to product that will end up meeting customer needs. |
| UE, Project lead | 7/9/2008 | At the meeting with the customer, communicated to customer that team is working iteratively & the UI will get refined over time. Customer was uncomfortable answering some questions b/c he is unsure of whether his answer is correct or not and b/c some aspects of the final end product are not clear. UE and project lead say that is why there is need to communicate with customer regularly. | Close collaboration with customer can catch potential issues/problems early on and save effort and lead to continuous improvement to product that will end up meeting customer needs. |

| | | | |
|-------------------------|-----------|--|---|
| Developer | 7/30/2008 | At meeting with customer, customer asks how <BMS> connection will work and how to prevent timeouts. Developer says once UI is touched, the connection will remain live for 20 minutes. Says everything will be handled through core connector. | Importance of meetings to identify and minimize misunderstandings among team members. |
| UE, Developer | 7/23/2008 | At the touchscreen meeting with customer, customer wonders who live connection to <BMS> will be maintained without timing out. UI depends on live connection. Developer will look into this. | Potential development constraint could affect UI design. |
| Usability engineer | 7/23/2008 | At the touchscreen meeting with customer, UE got feedback that it would be better to have pick list with text instead of typing in a number b/c text is easier to understand for end users. UE had though the text was hard to decipher so relied on numbers | Misunderstandings between team and customer will be resolved early with close collaborations with client. |
| Usability engineer | 7/23/2008 | At the touchscreen meeting with customer, UE got feedback that wording for a label should be 'operation complete' instead of 'job complete'. | Misunderstandings between team and customer will be resolved early with close collaborations with client. |
| Developer, UE | 7/23/2008 | At the touchscreen meeting with customer, workflow issue was resolved related to how data is input and when. Some jobs don't relate to components and need to input component history or input history comments (generally one or the other). | Misunderstandings between team and customer will be resolved early with close collaborations with client. This affects the UI |
| Developer, UE | 7/23/2008 | At the touchscreen meeting with customer, workflow issue was resolved related to how data is input and when. It is possible to have multiple notification histories for a particular notification. | Misunderstandings between team and customer will be resolved early with close collaborations with client. This affects both the underlying data model and the UI |
| Tester, Product manager | 8/13/2008 | At customer review meeting, Tester demos current implementation to customer. Customer has concern about real estate with a summary screen with a table of information. Product manager mentions that will hold usability evaluation to test that aspect of the UI. | Broader acceptance of usability and its use in validating the user experience goals of the system. |
| Product manager | 8/13/2008 | At customer review meeting, customer wants to know results of usability evaluation. How long did it take to input an entry? Product manager says the first run through was 3-4 minutes and the 2nd run through was 1-2 minutes. Says this demonstrates good learnability and efficiency. <customer representative> is happy because goal is to get it under 2 minutes. | Broader dissemination of usability and its use in validating the user experience goals of the system. Customer understanding of usability evaluations and user goals. |

| | | | |
|---|-----------|---|--|
| Project manager | 8/20/2008 | At customer demo, learned that can't use numpad popup for personnel ID b/c it's an alphanumeric entry. And can't use dropdown b/c workers move from work center to work center often & may not be updated in <BMS>. | This is work process information that affects UI design. Important issues that are caught early by interacting often with customer. |
| Project manager, UE | 8/20/2008 | At customer demo, there is question of whether to use text in a dropdown or an ID number. Since they have paper form still when doing their work and it has the ID/text on it. It may be faster for customer to just enter number. | This is work process information that affects UI design. Important issues that are caught early by interacting often with customer. |
| Project manager | 8/20/2008 | At customer demo, Disagreement between customer and new person from company that is seeing system for the first time. Should free entry be allowed in a field or should they be required to pick from a dropdown. This is issue of integrity of data (new customer) versus being able to input conditions that aren't in <BMS> (current customer) | Developing system with multiple customer contacts may result in higher applicability across the company but may not be as good a fit within a particular work center where it is used. |
| Project manager, product manager, usability engineer, developer | 8/25/2008 | At feature team meeting, discussion of whether to have on-screen keyboard or not. From UI perspective, should not have real keyboard since users showed down by switching from touchscreen to keyboard. Product manager does not want on-screen keyboard b/c already told customer there would be physical keyboard. Discussion is put off until can discuss with customer. | Discussion of features/changes without input from customer is risk that end product will not meet customer needs. |
| Developer | 8/27/2008 | At iteration meeting, Developer is not sure how customer has implemented <BMS>. Need to get the <BMS> expert from customer company to give them a copy of their database so can test from real data. | Poor communication/information sharing with customer may affect quality of end product and hinder collaboration. |
| Project manager, test manager, UE | 8/29/2008 | At daily meeting, proj. mgr, reviews work tickets and looks at the 'on-screen vs keyboard' ticket. Triggers another discussion about it w/ test manager involved (who hasn't been to meetings lately and is somewhat out of the loop). UE says should stop discussing issue and wait for input from customer. | Lot of discussion that could be resolved w/ customer involvement and communication. Speculation and assumptions about the system could lead to wasted effort. |

| | | | |
|-------------------------------------|-----------|--|--|
| Product manager, Usability engineer | 9/3/2008 | At iteration planning meeting, project manager asks if customer prefers physical keyboard or onscreen keyboard. Customer says he'd prefer physical keyboard. UE says usability studies show that on-screen keyboard would be best & expert evaluation says onscreen keyboard would be best. Customer says go ahead with onscreen keyboard--wants to see what it looks like. | Customer does not always know what is best when it comes to usability. Need for UE expert to ensure optimal user experience. |
| Project manager | 9/3/2008 | At iteration planning meeting, got feedback from customer about how to handle multiple time entries. | Highlights importance of regular customer contact to prevent drift between what customer wants and what is implemented. |
| Project manager | 9/3/2008 | At iteration planning meeting, project manager talks to customer about potential UI change related to failure code selections. One option will reduce change of bad data while other option will be more time efficient. Customer is confused b/c <BMS> data shown in demo does not look like customer's data. Turns out this isn't an issue b/c customer's data is different. | Some potentially big issues can be resolved with simple communication with the customer. Some things may turn out just to be misunderstandings. |
| Project manager, Usability Engineer | 9/5/2008 | At daily meeting, UE and PM have discussion of how data should be shown in dropdowns. They remember the conversation with the customer differently. UE says should just wait until talk to customer next. PM agrees. | Lot of discussion that could be resolved w/ customer involvement and communication. Speculation and assumptions about the system could lead to wasted effort. |
| Project Manager | 9/10/2008 | At iteration meet with customer, project manager asks customer about going up to site to do evaluation with real mechanics. Customer says mechanics don't know about it yet, but he'll "make them aware" | Limited access to end users has resulted in less optimal feedback about the usability of the system. Company culture and politics may further hinder access to them. |
| Usability engineer, developer | 9/10/2008 | At iteration meet with customer, Developer shows customer on-screen keyboard. UE gives summary of user testing results saying on-screen keyboard would be best. Customer says OK. | Open communication among team members will improve efficiency and promote more effective collaboration. |
| Developer | 9/17/2008 | At iteration meet with customer, developer got clarification on functions that will be delivered for Phase II. | Importance of open communication with customer to highlight holes in requirements. |
| Product manager | 9/17/2008 | At iteration meet with customer, Product manager demoed Phase II prototype to customer to brainstorm what new requirements & changes need to be made. | Importance of open communication with customer to highlight holes in requirements. |

| | | | |
|--------------------|-----------|---|---|
| Usability engineer | 9/17/2008 | At iteration meet with customer, there is push to add complexity to activity type field to make it cover more boundary cases so system will be more broadly applicable to other customers. UE asks if can push this to supervisor piece so only 1 supervisor has to worry abt it instead of 20 mechanics. Customer agrees--goal is to make it as easy as possible fo mechanics. | Importance of using high level goals to drive design decisions. |
|--------------------|-----------|---|---|

Cohesion-coordination incidents

| Team Cohesion (How well team is working together) | | | |
|--|--------------------------|--|---|
| Role | Date of incident: | Event description | Effect of incident |
| QA | 6/18/2008 | QA person (<qa consultant>), floated idea of using modeling and simulation software on project to model workflows at the plant to identify bottlenecks and help come up with most time-efficient work processes.. | Demonstrates team cohesiveness and willingness of team members to help each other in achieving goals of project. |
| Project manager, Product manager, Usability | 7/2/2008 | Product manager was supposed to get in touch with the customer to set up a meeting to discuss open issues the team put together since the site visit. As of the daily meeting, the product manager has not gotten back to the team about the meeting. | this relates to the product manager not attending the daily meeting. Inadequate communication with team members will hinder collaboration and in this case further delay the project. |
| QA manager | 7/14/2008 | At daily meeting, Qa manager mentioned that we should know what is required for us to release this later down the line if it becomes productized. (e.g. build machines, required software that needs to be installed, etc). We shouldn't let customer know we're delivering something w/ <web technology> & then fidn out later that we can't do it. | Blocking issues not related to development/usability within the team can hinder project velocity. |
| Development, Project manager | 7/28/2008 | Developer is trying to save effort later by planning ahead (do hard work now, so won't have to do as much later) as a way to manage risk. Project manager wants to develop incrementally only what is requested of customer now & show it to them asap as a way to manage risk. | Dev. is not willing to work in an 'agile' fashion- which has affected project velocity b/c of big up-front costs associated with how he's done work. |

| | | | |
|---|--------------------------|--|---|
| Developer, Project manager | 8/11/2008 | Developer hesitant to estimate time to complete development task (popup data entry controls). Does not want to 'promise anything'. Project manager wants to know if it will be done by Wednesday for meeting with customer. | Team members need to estimate development tasks so project manager can monitor progress and plan project. In agile project, estimations are key for planning iterations and measuring progress. |
| test manager, usability engineer | 8/29/2008 | At daily meeting, test manager, who hasn't been to a meeting in a few weeks due to other commitments, comments on how much work has gotten done since she last saw the system. "Excellent job guys". | Project velocity is relatively high at this point in the project. |
| Project Manager, Usability engineer | 9/10/2008 | At iteration meet with customer, Customer asks about Phase 2 UI. <usability engineer> quietly asks Project Manager if they can show her mockups for phase 2 to customer. Project manager says no. He doesn't want to show mockup to<customer representative>and says they'll probably do UI within Meridium desktop client | Differences of opinion between team members can result in giving customer inconsistent information. UE was not aware that they were doing Phase 2 UI in Meridium as she is only there part time. |
| | | | |
| | | | |
| | | Meeting Management | |
| Role | Date of incident: | Event description | Effect of incident |
| Project manager, Product manager, Usability | 6/13/2008 | Project manager scheduled the requirements meeting before the usability meeting. Usability engineer (through an email) stated that she wanted to discuss usability issues before talking about general requirements. Project manager ended up combining both meetings into one. | Open communication among team members will improve efficiency and promote more effective collaboration. |
| Project manager, Qa manager | 6/13/2008 | QA manager noted that the previous two meetings ran over and that that should be become a habit. This will cause other scheduling conflicts and frustrations among members of the team. | Purpose of daily meetings is to quickly review status among team members with tasks being done within an iteration. Having them run long and get off topic will frustrate and bore members of the team |
| Project manager | 6/13/2008 | Project manager ran the meeting with a set agenda and reviewed 'action items' that are currently on the portal. The meeting ended right on time. | Purpose of daily meetings is to quickly review status among team members with tasks being done within an iteration. Having them run long and get off topic will frustrate and bore members of the team. Using an agenda/portal helps guide meetings and makes sure it runs on time. |

| | | | |
|--|-----------|---|---|
| Documentation, Developer, QA | 6/26/2008 | The requirements review meeting was intended to review the user interface and software architecture--1.5 hr meeting. There was limited involvement from the documentation people there and from QA. The meeting delved very deeply into details of the architecture (developer and proj. lead) which was of limited value to certain members of the team. | Although having everyone in the team involved in what everyone else is doing, it is important to have everyone have a good high level view of what everyone else is doing without wasting people's time on unimportant details. This will make team members less annoyed while still keeping them involved. |
| Project lead, usability engineer | 7/16/2008 | At the daily meeting, before UE can demo screens, Project lead wants her to try screen sharing software so she can show screen from her laptop to another PC. This is to prepare for future meetings with AL. Took approx. 10 minutes of the meeting. | While useful for future meetings with customer, was generally not a good use of time for other team members. |
| Usability engineer | 7/18/2008 | UE expressed concern about time wasted in meetings that run too long and that certain people on the team don't have to be at. This has not been communicated to the project manager. | Open communication among team members will improve efficiency and promote more effective collaboration. |
| Usability engineer, Project manager, QA manager, Developer | 7/28/2008 | At daily meeting, Developer demoed current implementation. QA manager had a question of how best to show the data on a particular field. Lengthy discussion between Proj. mgr, Developer, QA manager about how to display data. UE says she will discuss this with developer later outside of meeting. | Team tends to focus too much on details during demos that are shown at daily meetings. Can make meeting run for extended period of time and derail focus of meeting. |
| Project manager | 8/25/2008 | Feature team meeting ran over by 20 minutes. Product manager, testers had to leave early | Having meetings run over time may waste team members time and make them feel bored and frustrated |
| Product manager | 8/27/2008 | Iteration meeting, Didn't meet with customer this week as he is off. Purpose of meeting is no longer clear. (Product Mgr. asks what are we doing here?) | Not having regular contact with customer may introduce risk of development drifting from what customer needs/wants. |
| Project manager, developer, product manager, usability | 9/19/2008 | At phase I wrapup, reviewed open work items. Dev. says all tickets are deferred to Phase II since Phase I is over today. Product manager agrees & asks what purpose of meeting is. Project manager says it's to verify what's deferred & what's left out of phase 2. | Differences among team members about how project should be managed. Perception that meeting does not have purpose leads to discord and thinking that their time is being wasted. |
| | | | |
| | | Project participation/attendance | |

| Role | Date of incident: | Event description | Effect of incident |
|--|-------------------|--|--|
| QA, Documentation | 6/9/2008 | At the iteration 0 meeting, we presented the responsibilities of each role (dev, qa, doc, usab., etc). QA and Doc managers stated that they will not be able to work incrementally as requested. There are limited resources that will make it hard to commit dedicated QA and Doc. person to the team. Doc. person wants to keep doing things the way they have been done in previous projects. | Inability or unwillingness to collaborate due to resource limitations will hamper how well the team develops the product (wrt testing and documentation in this case) |
| Product manager | 6/11/2008 | At standup meeting, purpose of meeting was to discuss requirements of the proposed project. It was expected that <product manager> would be at the meeting but he did not show up. | Limited commitment from some team members will slow the team down and hamper communication. |
| Product manager | 6/16/2008 | Product manager did not attend daily meeting. Project manager mentioned that there is a site visit scheduled for next Tuesday but did not have additional details (as product manager was not there) | Having team members that do not attend regular daily meetings hinders communication of progress/work and could hinder project progress. It could also lead to misunderstandings. |
| Product manager | 6/20/2008 | Product manager did not attend daily meeting. So could not participate in discussions about planning for site visit next week. | Having team members that do not attend regular daily meetings hinders communication of progress/work and could hinder project progress. It could also lead to misunderstandings. |
| Usability engineer, Project lead, product lead | 6/23/2008 | At meeting with VP of development, we find out that product lead has put together a draft agenda, but he has not shared it with the other people going on the site visit. | Not sharing information and being open hinders collaboration and threatens relationships/trust between team members. |
| Developer | 6/23/2008 | At the daily meeting, developer showed limited interest in participating in the discussion about planning for the site visit. Seems to want something to work on. | This problem relates to the fact that the project does not have customer involvement right now and didn't get it early on. Developer is not showing interest in participating now which may affect how architecture/system is designed later on. |

| | | | |
|--------------------------------|-----------|--|--|
| Product manager | 6/26/2008 | Limited involvement from the product manager at the requirements review meeting. Appeared to spend much of meeting working on other things on his laptop. | Problem of having team members working on multiple projects simultaneously means that they will not be able to devote enough attention to the project at hand and could threaten project velocity. This is especially true of the product manager, who is working on a very large number of projects simultaneously. |
| Product manager | 6/27/2008 | Product manager did not attend daily meeting. | Having team members that do not attend regular daily meetings hinders communication of progress/work and could hinder project progress. It could also lead to misunderstandings. |
| Product manager, Documentation | 7/2/2008 | Product manager and documentation did not attend daily meeting. | Having team members that do not attend regular daily meetings hinders communication of progress/work and could hinder project progress. It could also lead to misunderstandings. |
| Product manager, QA manager | 7/16/2008 | Product manager and Qa manager did not attend daily meeting. QA manager had another meeting to attend. | Having team members that do not attend regular daily meetings hinders communication of progress/work and could hinder project progress. It could also lead to misunderstandings. As long as we maintain contact with customer, reduced role from Product manager is acceptable. |
| Product manager, Documentation | 7/18/2008 | Product manager and documentation did not attend daily meeting. | Having team members that do not attend regular daily meetings hinders communication of progress/work and could hinder project progress. It could also lead to misunderstandings. |
| Product manager, Documentation | 7/21/2008 | Product manager and documentation did not attend daily meeting. | Having team members that do not attend regular daily meetings hinders communication of progress/work and could hinder project progress. It could also lead to misunderstandings. |
| Product manager | 7/23/2008 | Product manager did not attend daily meeting. Project manager mentioned that there is a site visit scheduled for next Tuesday but did not have additional details (as product manager was not there) | Having team members that do not attend regular daily meetings hinders communication of progress/work and could hinder project progress. It could also lead to misunderstandings. |

| | | | |
|--|--------------------------|---|--|
| Product manager | 7/25/2008 | Product manager did not attend daily meeting. Project manager mentioned that there is a site visit scheduled for next Tuesday but did not have additional details (as product manager was not there) | Having team members that do not attend regular daily meetings hinders communication of progress/work and could hinder project progress. It could also lead to misunderstandings. |
| Developer, project manager | 7/28/2008 | At TS feature team meeting, developer missed beginning of meeting. Halfway through meeting, PM called dev to bring him down to the meeting | Having team members that do not attend regular daily meetings hinders communication of progress/work and could hinder project progress. It could also lead to misunderstandings. |
| Project manager, product manager, tester, usability engineer | 8/18/2008 | Daily meeting started 10 minutes late. Usability engineer, tester, product manager sat and waited. Project manager was outside talking to someone else. | Non-timeliness of meetings may annoy team members and wastes their time. |
| Documentation | 9/8/2008 | At Feature Team Meeting, first appearance by documentation person in several weeks. Says has been busy on another project. | Without regular attendance to meetings, team members will not have an idea of the project or how it is progressing. |
| Documentation, Product manager | 9/19/2008 | Documentation asks how help will be integrated into product. I.e. if there are help buttons on each page in the UI or if there will be some separate help system. Product manager says product doesn't need much documentation--probably something as simple as a printed 'help card' | Question of how best to integrate documentation into agile usability development process. |
| | | | |
| | | | |
| | | SE+UE synchronization | |
| Role | Date of incident: | Event description | Effect of incident |

| | | | |
|--|-----------|---|--|
| Project Lead, Product manager, usability eng | 6/19/2008 | At phone conference meeting to plan for site visit next week, there was a difference between what product manager wants to bring to the meeting and what usability engr wants. she doesn't want to start mockups yet while project lead and product manager want to bring some mockups to show them. Product manager seems hesitant to present new/different ideas to customer. He seems to want to limit the scope rather than work with the customer to set the scope of the project. (keep customer in dark about certain things). | Misunderstanding between team members was identified and (mostly) resolved. Decided to mock up different solutions and present them to customer to jump start discussions. Product manager's hesitance to communicate openly with customer could potentially be issue. |
| Project lead, usability, QA manager | 7/7/2008 | At iteration planning meeting, Project lead wants to fit all work for a feature into a single iteration (design, development, QA, UI testing) QA manager and usability expert see difficulties in fitting everything into two weeks. | Project lead seems to not have clear understanding of agile and how even though work proceeds in iterations, changes can affect what was done in previous iterations. (e.g. usability evaluations, requirements changes, refactoring, etc) |
| Project lead, developer, usability, QA manager | 7/7/2008 | At the iteration planning meeting, there is disagreement about what should be done in the first iteration. Developer thinks we should have rough first pass at the whole UI. Team lead and QA manager talk about an incremental approach. | This reflects a difference of opinion on how the agile usability approach should work. Ideally, there should be a 'rough first' pass at most of the UI but there is an issue of how detailed that first rough pass should be. |
| Developer, usability engineer | 7/9/2008 | Right before meeting with customer, developer is hesitant to have navigation bar on the bottom of the page b/c he thinks there is potential problem w/ doing complex failure coding screen in the next iteration. | In an agile sense, designing everything ahead of time shouldn't be done, in a usability sense, it does matter. UE will do a low-level mockup of failure coding to see how it could work. |
| Project lead, developer | 7/14/2008 | At daily meet, Since team is not cleared to use <web technology> for development of the UI, project lead tells developer to focus on backend (talking to <BMS> database using Meridium <BMS> connector). | Scheduling of work for different team members will depend on outside factors. In this case, developer not cleared to work on system UI so works on backend. UE works on design. |
| Project manager, Usability engineer | 7/21/2008 | At daily meet (signaling end of iteration 1), project manager says need to lock down iteration 2 mockup so developer can work from a stable mockup. UE says she will lock down the iteration 2 mockup & work on iteration 3 mockup. | Properly coordinating work between UE and SE will ensure that project velocity is maintained and UE and SE are able to work without limiting the work of the other side. |

| | | | |
|-----------------------------|--------------------------|---|---|
| Project manager, QA manager | 7/22/2008 | At Iteration 1 Review meeting, QA mgr needs to know when the items in the plan will be delivered b/c Project mgr wanted QA work to be done within the same iteration. QA will not be able to do their work if most of the iteration is taken up with development work. They agree that there will be some overlap within an iteration. Development only has 1 week of development work in a two week iteration, then QA can work on testing that iteration's work. The next week, development can start work on next iteration's work and do rework they get back from QA. Project mgr. will add dates that each item will be delivered for <qa manager> so QA can start testing. | Still not clear how certain members of the team will work with the agile process being followed. Team has not come to any kind of equilibrium/regularity yet. |
| | | | |
| | | (SE+UE synchronization) Design+Implementation drift | |
| Role | Date of incident: | Event description | Effect of incident |
| Developer | 7/31/2008 | At daily meeting, Developer goes over work he's done. Developer implemented certain fields as text entry rather than as popup data entry fields like in the mockup. Developer does not believe popup entry fields are important and are focusing on other things. | Developer at times makes design changes as he implements the system without consulting UE expert. May result in compromises to usability of system. |
| Developer, Tester | 8/8/2008 | Tester created number of tickets based on comments from meeting with customer. Tickets include some wording fixes but some involve non-trivial change to UI. However, all tickets were assigned to developer rather than sent to usability engineer for review. This may result in 'fixes' that compromise overall usability of the system. | This is a communication problem. May result in UI fixes that are not tied to high level usability goals because fixes won't be discussed with usability engineer before hand. |
| Developer | 8/8/2008 | Developer added some animation effects to the UI to give feedback to user about things they have selected (e.g. bouncing button effect to show something is selected). This was done without consulting UI expert and was not shown to UI expert because she was not in the office the day he tried implementing it. | Making such changes to UI without consulting usability engineer may compromise overall usability of system for end users and effect the overall vision/consistency of the UI. However, such changes may have to be made is usability engineer is not available. |

| | | | |
|--|--------------------------|---|---|
| Project manager, developer, usability engineer | 8/15/2008 | At daily meeting, project manager shows UE the current implementation. UE mentions that the summary page does not match the original mockup. Developer made the summary table 'active' meaning that clicking on a line will bring up the update page for that entry. Also, the table shows more information than originally intended. Project manager says since the summary page is so close to being 'done' (i.e. it has been implemented), that any changes should just be referenced from the current implementation and not the original mockup. | UE was absent for most of previous 1.5 weeks b/c of conference and other issues. This has caused divergence between what developer implemented and what UE designed. (b/c UE wasn't here to answer questions, make clarifications, do mockups) etc. |
| Project manager, usability engineer | 8/15/2008 | At daily meeting, project manager shows usability engineer current implementation. (developer is currently out on vacation and usability engineer was out previous 1.5 weeks b/c of conference and other issues). UE sees the mockup--and the new popup data entry screens--and is surprised because it looks exactly like her mockup. | Project manager elevated priority level of popup data entry screens based on usability evaluation. UE is also surprised b/c developer tends not to implement mockups exactly as they are designed. |
| Project manager, developer, product manager, usability | 8/25/2008 | At feature team meeting, Project Mgr and Developer have discussion of how to handle free text in a picklist (pick a selection or enter text). Product manager suggests assigning it to the UI person to review--> backing up the UI person. | Acknowledgement of different areas of expertise that each member has will improve team collaboration. |
| | | | |
| | | | |
| | | | |
| | | (SE+UE synchronization) Usability Evaluations | |
| Role | Date of incident: | Event description | Effect of incident |

| | | | |
|--|-----------|---|--|
| Project manager, Product manager, usability engineer | 7/22/2008 | There is issue of how to validate UI. Product manager says it may be possible to get other people at customer site to try out UI and see if it works. Project manager says it won't be possible to go up to <customer company> every week and wants to do a remote evaluation tomorrow b/c he wants to get feedback as soon as possible so don't waste development effort. UE says she'll be ready next week, will need time to prepare evaluation. There will be some changes made but it's not a waste (it's like refactoring-this is agile). | There is some miscommunication here between UE and Project manager. Project manager wants quick evaluation with someone at customer site acting as end user while UE wants more well planned evaluation with actual end user at customer site. |
| Project manager, usability engineer | 7/31/2008 | At daily meeting, project manager asks if it will be confusing to have radio buttons look the same as regular buttons. UE says she will validate risk in next user study (claim). | Claims (which are used in UE evaluations) are used to record team member concerns and save them for validation at evaluations (rather than argue and make change right then). |
| Project manager, developer, UE | 8/25/2008 | At daily meeting, developer says he will estimate development effort for changes that UE put into system based on the usability studies. UE gave priority for the changes. Project manager can then rank them based on severity of the problem and the estimated development effort. | Syncing of information from UE and SE can aid in making prioritization and development decisions. |
| Project manager, usability engineer | 9/8/2008 | At feature team meeting, project manager asks UE if it's really necessary to go up to customer site to run evaluation or if evaluation can be run remotely. UE says won't be the same since she won't be there to see contextual stuff from users (gestures, hesitations, etc). | Team has limited understanding of what UE person does and why. Open communication channels are critical to resolve these problems. |
| Usability engineer | 9/8/2008 | At daily meeting, UE gives quick overview of expert evaluation of the touch screen with the touch screen expert at the company. Says it went well overall. | Broader dissemination of usability and its use in validating the user experience goals of the system. Team understanding of usability evaluations and user goals. |
| Product Manager, usability engineer | 9/8/2008 | Product manager sends email saying he's reviewed touch keyboard and thinks it might be better to just use all caps on the keyboard. Doesn't explain why this is the case. UE sends email response saying she ran user studies and no one mentioned that as a problem and studies show using all caps is disruptive to users. | UE can give rationale for why design should remain as it is. Has evidence and acts as the user advocate. |

| | | | |
|---|--------------------------|--|---|
| Product Manager, Usability Engineer | 9/13/2008 | At daily meet, prod. mgr. says he had an issue with not having a 'cancel' button on the popups. UE says that may need little red x button to cancel out of popups. Says during usability tests, some people didn't realize how to cancel out of popups. | Usability tests reveal new unforeseen problems that need to be resolved. |
| | | | |
| | | | |
| | | UI affecting implementation and implementation affecting design | |
| Role | Date of incident: | Event description | Effect of incident |
| Developers | 7/2/2008 | At daily meeting, developer consultant (<developer 2>), said it was possible to create rich UI with <web tech> but project lead and dev lead have limited experience with <web tech> and are concerned with scheduling. This is in response to the new requirement that the system be a web app. | Development limitations and time constraints may limit what can be achieved in terms of user experience. |
| Usability engineer, Developer | 7/9/2008 | At UI demo meeting, UE presented lo-fi mockups developed in (power-point equivalent). It included 'circle-shaped' buttons. Developer mentioned technical issue with using circle button-> requires use of 'pictures' instead of button controls. Would require someone to create the pictures in Photoshop. | presenting prototypes can uncover technical limitations which may affect how the UI is developed. |
| project lead, product lead, Developer, UE, QA manager | 7/11/2008 | At daily meeting, developer showed basic implementation he did of UE mockup using <web technology>. This was a code spike to see if <web technology> can be used. Benefit from developer/project lead's perspective is ease of development and improved look & feel of final end product. QA manager wants to know what cost is in terms of infrastructure needed at Meridium to run and test the application using <web technology> + <tech> and <>. Product lead wants to know what benefit is to the end user compared to cost of using it. | Technical issue which may improve ultimate usability of the end product but has additional potential costs/issues. There needs to be cost/benefit analysis to see if it's worth it. |

| | | | |
|--|-----------|---|---|
| Project manager, Product manager, Usability engineer | 7/22/2008 | At Iteration 1 Review meeting, Project manager goes over data model. UE sees that operation is not directly related to history. Asks if this will have implications for the UI interactions/workflows. In powerpoint that customer gave, both work order and op # were collected when history were entered. Gets a clarification from Product manager. UE says she will have to update the mockups based on the clarification. They'll discuss it in detail later at another meeting. | Proper coordination between different team members means that problems/clarifications can be found and addressed as they work together. |
| Project manager, Product manager, QA manager, Usability engineer | 7/28/2008 | At daily meeting, developer mentions that there may be development limitation to what the UI can do. Can't constantly poll <BMS> for data b/c it is too time-consuming. Right now, the developer implemented the system so the user is required to hit the 'enter' key before polling data from <BMS>. Team discusses different options. UE mentions that the mockup is different from the implementation and problem won't exist if it's implemented that way. | Development limitations and time constraints may limit what can be achieved in terms of user experience. |
| Developer, Project manager, usability engineer | 7/30/2008 | At meeting with customer, Developer mentions that one area of UI will need a 'save' button or 'saving' popup b/c may take time to communicate with backend. UE will discuss with dev outside of meeting. | Implementation constraints may affect UI design and vice versa. |
| Developer | 7/31/2008 | At daily meeting, at certain points of the UI when the system is accessing <BMS>, there is a pause so the system may seem sluggish or unresponsive to the user. Needs some type of indicator to show user system is working. | Technology limitations could affect how the UI is designed. |
| Project manager, developer, usability engineer | 8/27/2008 | At iteration meeting, Developer demos latest functionality w/ actual data retrieved from <BMS>. Discusses w/ project mgr & UE person potential problems w/ amount of data returned and how it affects the UI. E.g., returned list of a popup list is huge, may require change to UI. | Data, technical issues may affect UI. |
| | | | |
| | | | |
| | | | |
| | | Project visioning and goals | |

| Role | Date of incident: | Event description | Effect of incident |
|--|--------------------------|--|--|
| QA manager, Usability, Project manager | 6/23/2008 | At daily meeting, Usability person presented 3 different workflows for different possible design options. QA manager is worried about setting expectations of customers too high by presenting options that we can't actually deliver. Project manager just wants to use potential workflows as 'strawmen' to get discussions going to customer. | Potential disagreement about what should be presented to customer (different design options). This one relates more to feasibility of implementation more than keeping the customer in the dark and was resolved in the meeting. |
| QA, Usability | 6/23/2008 | At daily meeting, usability person presented 3 different workflows for different possible design options (root concept claims). She also presented upsides/downsides of each approach. QA person (<qa consultant>) suggested that she also show upsides/downsides of the existing approach so customer can see benefits of each approach over their existing approach. | Useful feedback provided by other team member at daily meeting shows usefulness of reviewing such work products with other team members. |
| Project lead, usability engineer | 7/2/2008 | At daily meeting, Project lead presents list of high level usability goals prepared by usability expert. Project lead also wants to add meridium goals and other project goals to this list so everything can be prioritized. | Gives the team a better understanding of project goals from user's perspective. |
| Usability | 7/7/2008 | At the daily meeting, usability expert presented a 'quick usability' document that summarizes high level goals, user descriptions and high level scenario. Includes links to more detailed usability documents. | Gives team a better understanding of project goals and high level usability goals. |
| QA manager, product manager, project manager | 7/31/2008 | At daily meeting, QA manager and Product manager want to code for 'localization' now. Project manager tries to explain that it's not currently a requirement & we need to focus on delivering what customer wants right now (high priority stuff). | Some team members do not understand/buy into certain concepts of agile such as JIT requirements analysis, and not designing for the future. |

| | | | |
|---|-----------|---|--|
| Project manager, Product manager, developer, Usability engineer | 8/27/2008 | At iteration meeting, there is tension between tailoring a solution for customer & having the system be generalizable. Project manager says to tailor it for current customer now to provide best possible experience for them. Dev. says risk is that any future fixes will be trailored for customer & will be hard to generalize later. Project manager says extra effort is about the same since extra effort would be required now to make it generalizable and end result would be less usable. | Not having consensus on high level goals for project may result in end product that doesn't meet any goals satisfactorily. |
| Product manager, Usability engineer | 8/27/2008 | At UI meeting, UE presents UI of phase II. Product mgr notes that it includes much more functionality then expected. (Customer only asked for edit & accept functionality). Prod. mgr, doesn't want to present additional functionality to customer b/c they will expect it then. Doesn't want to present functionality they didn't pay for already. | Tension between business needs of development company and meeting needs of customer/end user. |
| Project manager, Usability engineer | 9/5/2008 | Proj. Mgr. says maybe it would be better to show data in a particular way b/c will be easier to productize the system later for other customers. UE says she's confused b/c sometimes he says to design for current customer and sometimes he says to generalize for others. | Inconsistent focus on particular high level goals may result in UI that will not meet needs of end users/customers and lead to a UI that has inconsistent high level design. |
| Product manager, Usability engineer | 9/8/2008 | Product manager discusses maybe getting rid of auto-field popup so it's more understandable. UE agrees and resulting UI will have higher comprehension although time to completion may be slightly slower. | Consideration of high level usability design goals to drive design decisions will ensure overall vision for system is clear. |
| UE | 9/10/2008 | Customer is hesitant to have brand new Phase 2 UI b/c he wants to have query functionality of Meridium Client. Thinks standalone would have less functionality and cares less about usability for supervisor. | Difference of opinion between usability engineer and what customer wants. UE sometimes does not consider outside factors that may affect design. |
| Developer, Product manager, Usability engineer | 9/17/2008 | At iteration meet with customer, Developer and Product Mgr. say they put together supervisor piece in Meridium Client. UE not happy with it--inconsistent view between kiosk & supervisor piece. Product mgr. wants it this way b/c of cost/time savings. (UE had no impact on Phase II design.) | Difference of opinion between usability engineer and what customer wants. UE sometimes does not consider outside factors that may affect design. |

| | | Prioritization of tasks based on different factors (iteration planning) | |
|--|--------------------------|--|---|
| Role | Date of incident: | Event description | Effect of incident |
| Project lead | 7/9/2008 | At iteration planning meeting, Project lead presented initial iteration plan with customer. To confirm what is being delivered and when (relative priorities) | Perhaps showing entire working iteration plan was too much information for customer, but it is important to communicate what is being devliered and when. |
| Project Manager | 7/18/2008 | Project manager uploaded updated iteration plan with time estimates for the developer. VP of Development wanted cost info. to help with contract they're working on for <customer company>. Also says he will add UE and QA and Doc estimates for their tasks. Each person will also add in their tasks to the iteration plan. | Feature-level plan for all the project participants will give others on the team an understanding of what everyone else is working on and will allow Project Manager to track project velocity. |
| Project Manager, Developer, Usability engineer | 7/21/2008 | At daily meet, project manager reviewed work done by everyone by looking over iteration plan, first developer and then UE. Also sets work items for next few days. | Feature-level plan for all the project participants will give others on the team an understanding of what everyone else is working on and will allow Project Manager to track project velocity. |
| Project manager | 7/22/2008 | At Iteration 1 Review meeting, project lead reviews work done in first iteration by going over edited iteration plan--added work item numbers for each relevant item on the list. | Will allow team members to more easily track status of features (whether they are complete). Likely most useful for QA and Doc b/c of their reliance on work items in their work. |
| Developer, QA Manager, Project manager | 7/25/2008 | At daily meeting, developer says that he has fallen behind in development tasks. Developer has worked on tasks out of priority order. QA manager says need to re-estimate tasks and 'be realistic'. Project manager says if work items aren't done, the rest of team needs to know it. | Not sharing information and being open hinders collaboration and threatens relationships/trust between team members. |
| Developer | 7/25/2008 | At daily meeting, developer says that he has fallen behind in development tasks. Developer is slowed by fact that he is using new, unfamiliar development technologies. Says he may need someone else to work on backend stuff--needs more development support. | Lack of development resources may hinder project velocity. |

| | | | |
|--|-----------|---|--|
| Qa manager, Project manager | 7/28/2008 | At TS feature team meeting, project manager shows current project plan. QA manager still not clear on how and when things will get delivered now that development has slipped. PM explains that schedule has slipped 1 week so all items will be devliered one week late. | Team is still having difficulty following project progress/seeing project velocity. These issues need to be resolved in later iterations. |
| Qa manager, Project manager | 7/28/2008 | QA manager points out that iteration 3 has 82 hrs of work planned for one developer in 6 days. Says it is doomed to failure and notes that should not asusme 100% efficiency (b/c of lunch, bathroom breaks, etc). PM says plan will be updated. | Team is still having difficulty following project progress/seeing project velocity. These issues need to be resolved in later iterations. |
| Project manager, product manager, developer | 8/11/2008 | At daily meeting, Project manager wants developer to work on on-screen data entry popup controls--as they are limiting usability goal of efficiency so they can be shown to customer at Wednesday meeting. Developer mentions that he may not be able to get it done by then. Developer and product manager want to show customer older, stable version rather than new untested functionality. | Certain members of the team still have a mentality to shield the customer from 'unfinished' functionality. Don't view customer as a member of the team. This could hinder communication with customer and overall collaboration. |
| Project manager, developer, usability engineer | 8/11/2008 | At daily meeting, Project manager says he received email from <usability engineer> saying popup stuff is critical thing to get done based on usbility evalautions. Not having the on-screen popup controls for data entry requires user to switch repeatedly from screen to keyboard, resulting in errors and reduced time to completion. Developer mentions that should be able to get it done in the next two days. | Usability results can affect priority of development tasks. |
| Project Manager, Developer, Usability engineer | 8/18/2008 | At feature team meeting, Developer shows excel sheet of work items that he still needs to work on. Project manager expected him to update the work items in Visual studio so he could develop a plan from there. | Improper use of tools by team members can make it more difficult to manage project. |
| Project Manager, Developer, | 8/18/2008 | At feature team meeting, Project manager went with the team and looked at the list of open work items and prioritized fixes to do before demo w/<customer representative>on Wednesday (based on Developer's estimates & importance of fixes--especially ones from customer b/c team wants to be responsive to his comments) | Proper prioritization of issues with customer is critical to successfully meeting most important customer needs |

| | | | |
|---|--------------------------|--|---|
| Project manager, UE, Developer, Product manager, tester | 8/25/2008 | At feature team meeting, Prioritized work items based on input from team members (severity of problem, time to fix problem). Input from product manager w/ respect to meeting customer/business needs, input from UE wrt usability of end product and input from developer wrt functional correctness. | Close collaboration among the team with input from all members can help in making decisions in how to prioritize work. |
| Project manager, developer | 8/25/2008 | At feature team meeting, project manager has discussion with developer about him working on non-essential items (researching other things, upgrading merid. DB), instead of working on critical work items. Developer's work priorities need to be clear. Developer doesn't think he's ever gotten sidetracked. | team members not seeing eye to eye on certain issues may hinder collaboration. |
| Usability engineer, Project manager | 9/8/2008 | At daily meeting, UE says can defer work item related to 'wait' animation. Found that no user so far has had issue with it. Will wait until evaluation w/ actual end users. Proj. mgr defers ticket until next phase. | Development tasks can be saved based on results from usability evaluations. |
| Project Manager | 9/8/2008 | Project manager has features ranked by 'product priority'-importance in terms of something that needs to be delivered to customer. And ranks them by 'planning priority' which is ranking based on product priority and with scheduling/development resources issues taken into account. Ranks items by planning priority and reviews them with developer. | reviewing development priorities on a regular basis can help team get understanding of project progress and velocity. |
| Project Manager, Usability Engineer, Developer | 9/13/2008 | At daily meet, usability engineer gives relative priorities of usability fixes with respect to high level project goals so project manager can integrate into schedule. (high, med, low). Developer gives development estimates for each of the usability fixes. Can decide which things to fix now and which to defer until phase 2. | Usability engineer, and developer can give project mgr. enough information to make decision related to which usability fixes to implement |
| | | | |
| | | | |
| | | | |
| | | | |
| | | Information sharing | |
| Role | Date of incident: | Event description | Effect of incident |

| | | | |
|----------------------------------|-----------|---|--|
| Project lead, developer | 7/2/2008 | At the daily meeting, project lead presents system architecture document prepared by lead developer. It's already a little out of date b/c of the new requirement that it be a web application. Gives team an idea of how <BMS> data is stored and how it relates to workflow. | Gives team better understanding of underlying architecture and what customer is trying to accomplish with <BMS> through meridium. |
| Project lead, usability engineer | 7/7/2008 | At the daily meeting, usability expert presented scenario of phase 1 functionality. Had details of UI that led to team discussion about specific design options. Project lead suggests to keep scenarios general to so can focus on the task flow rather than details. | Need to manage team expectations and have them focus on the right things. In this case, intention was to have them focus on the basic workflow in the scenario, not the UI details. |
| Project lead, usability engineer | 7/9/2008 | At UI demo meeting, project lead wanted to edit & play with prototype but UE uploaded uneditable version. UE then uploads word version so he can play with it. | Team members should upload editable versions of documents to portal to facilitate collaboration. BUT, UI design is primary responsibility of UE and there is risk of 'design by committee' |
| Project lead, Usability engineer | 7/16/2008 | At the daily meeting, Project lead goes over the scenarios that UE developed for the first iteration. UE asks who will use/look at scenarios she put together. <project manager> says testers will use scenarios to plan tests and developers should look at them too to get better understanding of the project and how it will be used. Also, work items will be tracked against the scenarios to track project progress. | UE person is concerned that her design artifacts are not being reviewed by the team and are not sure how other team members will use/review them. Project lead tries to make clear that they will be useful to other team members. |
| Usability engineer | 8/27/2008 | At iteration meeting, UI person notes that never got a set of copies of the data entry form from the customer. These forms would be used to see what kind of data needs to be entered and what it looks like. | Poor communication/information sharing with customer may affect quality of end product and hinder collaboration. |
| Usability engineer, Tester | 9/10/2008 | Customer says phase 1 UI looks good and wants to focus on phase 2 now. Says he'd like to see what is being planned for phase 2. UE agrees--> saying mockups will help spark discussion. Tester agrees, says often customer would not know what he wanted until we showed something. | Importance of mockups in showing potential designs and spark discussions about requirements with customer. |

Appendix B. Retrospectives from case studies

Web Components Retrospective

4/09/08 Issues identified in Retrospective and Proposed solutions

Issue: How do we disseminate information about UI design to the entire team? <usability engineer> and <product manager> hold an extended UI meeting on Mondays but the entire team is unable to attend.

Proposed resolution: After the UI meeting on Monday, <usability engineer> and <product manager> will touch base with <offsite dev lead> to review the feasibility of implementing proposed UI changes. Then <usability engineer> will upload the UI changes to the portal along with a list of changes with related design rationale. <product manager> will then present these changes at the daily meeting on Tuesdays.

Issue: How are small UI changes communicated to developers so they are actually implemented by the developers?

Proposed resolution: If the proposed changes are a part of the current iteration (i.e. they have not been delivered yet), then they can be communicated informally to developers. If the changes relate to features that have already been delivered, then they need to be written up as tickets. <usability engineer> will work with <documenter> on this.

Issue: How to keep track of important updates to the Portal so action items and open issues are addressed at some point?

Proposed resolution: Once a week (at the feature team meeting?), the team lead needs to present any messages & action items that have been posted to the Portal so they can be reviewed and resolved. Also, if anything important is updated/uploaded to the Portal, then that person should send an email out to the Web Components team so they are aware of the change.

Issue: How are non-development oriented tasks assigned and managed by the team?

Proposed resolution: There is currently an 'action items' section on the portal. The team lead maintains the list of action items that come out of the daily meetings will review this list on a regular basis—probably at the weekly feature team meeting.

Issue: Some team members including <documentation mgr> and <usability engineer> are unable to attend every daily meeting and therefore do not have complete awareness of project progress.

Proposed resolution: The following are proposed foci of the Web Component meetings throughout the week.

- Monday: Iteration review meeting—The developers will demo all work done in the previous iteration to give the team an overview of the work that was just delivered. This meeting will be especially helpful to team members who are unable to attend all the daily meetings.
- Tuesday, Thursday, Friday: Daily standup meetings—These are daily status meetings where team members present work that was done the previous day, tell what work they will have done by tomorrow, and describe any problems/issues that are preventing them from doing their work.
- Wednesday: Feature team meeting—In addition to the daily standup meeting, this meeting will include a retrospective where team members reflect on the past sprint and identify and process-related problems that need to be resolved. This should also involve revisiting issues identified in the previous retrospective that have not yet been resolved. This meeting will also be used to discuss ‘miscellaneous’ project issues that do not fit into the other meetings.

Issue: Some members of the team don’t have good awareness of overall project progress (e.g. What features have been implemented, what features are going to be worked on, what features have been deferred and how will that affect overall progress?)

Proposed resolution: On Fridays at the end of the iteration, <offsite dev lead> and <project manager> will plan the next iteration based on the ‘Web Components Iteration Plan’—which details project-level details across all of the planned iterations—and on progress the team has made in the last iteration. On Sundays, <offsite dev lead> will meet with <offsite company> developers to come up with the Backlog for the upcoming iteration. Both the iteration backlog and the high level iteration plan will be updated and uploaded to the Portal by Monday. These items can then be reviewed at Monday’s daily meeting.

Touch Screen Retrospective

What went well

- Incrementally showing progress to customer (through mockups, code, etc) was valuable since customers can’t think in terms of requirements.
- Incremental development led to incremental development of our own knowledge about the system. We were able to catch issues early.
- Incremental development supports refactoring of code; Easier to refactor code because it’s a smaller change.
- Incremental mockups allowed us to show customer that his feedback got reincorporated into product. (Customer feels ownership of product)
- Showing mockups early was good way of getting requirements.
- Showing mockups saved development effort later and allowed developer to learn new

technology early.

- Clearing up misconceptions and misunderstandings by talking with customer was important.
- when team members were collocated, we were able to have ad hoc meetings to clear up misunderstandings quickly.
- When team members were collocated, we were able to fix certain problems without the overhead of writing tickets.
- UI Design decisions could be guided by usability tests.
- UI Design decisions provided information that customer didn't necessarily know. (I.e. sometimes the customer didn't know if one design was better than another).
- Having usability engineer was had a huge positive impact. Allowed developer to focus on implementation/design issues instead of trying to handle UI issues.

What didn't go well

- Interface changed to the very end so documentation was unable to start their work. (How can documentation be integrated with constantly changing UI. Or can we freeze the UI at some point?)
- Team sometimes debated features too much based on assumptions and misunderstandings about what the customer might think.
- At times there was a difference of opinion among team members in terms of what to communicate with the customer. Team wasn't always on the same page before meeting with customer.
- Scope creep.
- Interface changes made it hard for QA to do their work. He would write test cases, but then UI would change so he would then have to redo work.
- Development often did not adhere to schedule. This does not refer to the project running late, but to functionality slipping to later iterations and to constantly changing requirements.
- Communication between development and usability was difficult at times. Expectations were different between the two.
- There were communication problems because the team did not have dedicated resources. (E.g. the usability person was part time)
- Was not clear when changes/fixes should be documented or not. (Especially problematic with a team without dedicated resources).
- Prioritization of features was not clear. Different team members had different ideas of what should be worked on when.
- Misconceptions between team members about what other team members wanted/thought.
- Team had limited experience/exposure with <BMS>.
- Sometimes customer did not have opinion about how something should be designed and team would have to guess.
- Externally imposed requirements caused problems. (E.g. there were debates on the product management side on whether the system should be <customer company> specific or should be generalized for future customers)
- Product management had limited information on the project—had no background research (outside of information from <customer company>) in terms of how the project should be developed.
- Didn't have access to real users to run usability tests on.
- Trying to fit usability tests into schedule so fixes can be fit into development schedule so they are timely was difficult.
- Having developer do task estimation during team meetings puts the developer 'on the spot'

and not all team members can contribute.

Fixes

- Interface changed to the very end so documentation was unable to start their work. (How can documentation be integrated with constantly changing UI. Or can we freeze the UI at some point?)
 - Documentation can get involved after major releases. But should have documentation present once a week to keep up with how the project is going.
- Team sometimes debated features too much based on assumptions and misunderstandings about what the customer might think.
 - Maybe meeting with the customer more than once a week would help.
- At times there was a difference of opinion among team members in terms of what to communicate with the customer. Team wasn't always on the same page before meeting with customer.
- Scope creep.
 - Need to carefully evaluate new requirements as they come in. Will the functionality provide value to customer and can we do it in the time available? Discuss these issues with customer and have them understand that they are part of the team. Manage expectations.
 - Have tradeoff information available to customer (why should this be done, how important is it, how much time will it take). When issues come up, team should make a decision based on importance/severity/time to complete before bringing to customer.
- Interface changes made it hard for QA to do their work. He would write test cases, but then UI would change so he would then have to redo work.
 - Have 'UI freezes' for a particular release. After several iterations, halt UI development so QA can proceed with testing. QA should not have tests that are so specific that they have to be rewritten. They should stay at a high level until a release. (Function level tests rather than 'click here, click there')
- Development often did not adhere to schedule. This does not refer to the project running late, but to functionality slipping to later iterations and to constantly changing requirements.
 - Estimates should include time to do things like 'writing unit tests' and doing QA testing. Keep the estimates separate so they don't get consumed by development time.
 - Development needs to start push for automated unit testing.
 - Don't assume 100% efficiency for the developer.
- Communication between development and usability was difficult at times. Expectations were different between the two.
 - Colocation would be important to address these issues. Maybe even sit team members next to each other.
- There were communication problems because the team did not have dedicated resources. (E.g. the usability person was part time)
 - Having a discussion board on the portal was helpful for previous project (web components)
- Was not clear when changes/fixes should be documented or not. (Especially problematic with a team without dedicated resources).
- Prioritization of features was not clear. Different team members had different ideas of what should be worked on when.
- Misconceptions between team members about what other team members wanted/thought.
- Team had limited experience/exposure with <BMS>.
- Sometimes customer did not have opinion about how something should be designed and team

would have to guess.

- Externally imposed requirements caused problems. (E.g. there were debates on the product management side on whether the system should be <customer company> specific or should be generalized for future customers)
- Product management had limited information on the project—had no background research (outside of information from <customer company>) in terms of how the project should be developed.
 - Getting information from multiple customers will be important for when the system is productized (but this is a tradeoff as you may get conflicting information).
- Didn't have access to real users to run usability tests on.
- Trying to fit usability tests into schedule so fixes can be fit into development schedule so they are timely was difficult.
- Having developer do task estimation during team meetings puts the developer 'on the spot' and not all team members can contribute.
 - Have developer do estimates before meetings and team can help with prioritization.

Tips for the future

- Develop relationship with customer early; Showing functionality in increments instead of showing whole thing at once—getting to know one another, how things will work; showing we can get things done; (customer confidence and our confidence has increased over time)

Appendix C. Case study interviews

Post Project Touch Screen Team Interviews

Developer 1

General questions

What was your role in the project? What did you do?

- I was the lead developer.
- I was also on the handheld project.
- I never worked on both at the same time. One of them ended and then one of them started. Not much overlap.

Project goals

What were the key design goals for the system?

- Ease of use, ease of deployment of the application itself. The reason we wrote it as a web application was so companies wouldn't have to install it on a machine. No setup required.

Do you believe everyone had a clear understanding of the goals? Were everyone committed to those goals?

- Sure. As far as phase 1, the team was focused on those goals. After phase 1 got completed, the goals themselves changed a bit. They changed from we had all the resources we need, to now we have to ship it so we have to make sure we don't overpromise to the customer. That's what we did in phase 1, we extended scope so much it became stressful to complete the project. Other goals emerged. Some things we took for granted such as, this app. Needs to run 24/7 but we've been seeing crashes. So now we've been spending effort on those things. Not having reliability as a goal hurt us.

Were there goals that conflicted with one another? How were they resolved?

- Yeah. In terms of designing the app., there were conflicts in terms of do we do what <customer company> wants or do it so other customers can use it. For example, we had the activity type screen but had to change it to a picklist so there could be many options downloaded dynamically from <BMS>. After phase 1, changed goal to make it more generalizable. UE was not involved.

Were these goals used to guide design decisions in the project?

- **Yes, definitely.**

Do you think the team met the project goals? Why do you believe that it has or hasn't?

- I think we've met all the goals in terms of the kiosk application itself. But in phase 2, we realized the supervisor piece was not going to be usable. We had a design from <usability engineer> that would be more usable but we had resource constraint so couldn't execute it. We are struggling with reliability but that wasn't a goal to begin with. (Now it's a goal though).

Did the team always have a common understanding of what things needed to be worked on and why?

- Oh no. The time was so constrained and not only that, because the designs were static designs in terms of mockups, it was impossible to consider what would happen to things when it became interactive. I had to make those decisions on the fly. And partly because <web technology> is a new technology I figured out better ways to do things so I had to refactor code.
 - o In regards to making decisions on the fly – no. <usability engineer> felt irritated that I did that, but I always discussed with her my rationale and I always got her opinion whether she agreed or not. If she had a compelling reason, we would have changed it back. I always discussed it with <usability engineer> and if it was horrible I would have changed it to something more suitable. If I could reach her, or it was a big decision I would talk to <usability engineer> first.
- As a team we were so busy with trying to determine the requirements, and trying to determine the requirements took the whole of phase 1. Up until the last week of phase 1 we were still discussing requirements.

Customer relationship

How important was it to be in contact with the customer/client during this project?

- I loved the contact. However, the contact was not handled properly. Because we started saying too many yeses to the customer's demands. We extended our scope so much (maybe by 50%) and that really resulted in me working that much more.
- Maybe we should have had more written documents about what exactly the requirements were and that should have been shared with the customer. So things weren't in flux. And we should have had someone that knew <BMS> from the get go.
- The things we discussed should have been in writing with them. And at the end of the meeting we should have updated a document that had the requirements.

What benefits were derived from interactions with the customer/client?

- We always kept expectations of the customer in check. We ended up with the most streamlined application because we didn't add unnecessary bells and whistles in there. And because the customer would agree with our design, that would give us more confidence.

How does the amount/quality of contact compare to other projects you've been on?

- It was 1000% better. On the handheld, we have a product manager but even our contact with the product manager was limited. In this project, we had direct contact with the customer and way more contact than the product manager on the other project. What keeps happening on the handheld is what frustrates me, there are too many throw away prototypes and the requirements keep changing all over board.

Did you feel that there should have been more/less customer contact? What should have been different with respect to this?

- I think if it was done right, we would have needed less contact but because we didn't do it right we had to have too much contact. I think we could have had less contact and that would have been fine. (Referring to nailing down requirements, having an <BMS> person on board).

Team Collaboration

How well did team members collaborate with one another? Did you notice certain problems related to how well one person worked with another? What caused these problems?

- Not too well. So we had no documentation support. Our QA was inexperienced. Our usability engineer and developer weren't working at the same time and place so that definitely had an effect. Also, the team lead did not manage the project. Deadlines would come and go and no one would track things that were said. And that would happen in front of the customer itself. Inexperienced team lead. And for me, this was the first project as a lead developer.
- Things were really bad at the beginning. From really, really bad, it came to bad. And I still think it's mediocre.
- Compared to the handheld team, in the handheld team the team lead has a handle of every issue we have. That team lead manages several projects. Also, it has proper QA and documentation support so we can get their feedback in and that helps a lot. Especially with the .<newer> release cycle it was like clockwork. The plan was laid out and it was executed with precision.

What sorts of problems were encountered and what caused them?

Did you ever feel that certain team members were not involved enough or that they were overstepping bounds?

- Yeah, I think the problem was the lack of involvement almost all members. I mean we didn't have any documentation support. QA was inexperienced and didn't know any better and because of that it was a lack of involvement. And our UE was not a full time employee, even though she did a great job, I was forced to step into her boundary which if I didn't have to do I wouldn't have done it. And also, it felt like there was a lack of involvement from the team lead. Deadlines weren't enforced. Issues weren't tracked—like every time we start a project we create work items for features we're going to work on. We didn't have those work items. I was too busy trying to code the project, I couldn't spend much time on the process. But then again I wasn't asked to do things I should have done. And normally on the handheld team, if you don't do something, the team lead realizes it and tells you to do it.

Were you typically aware of what other team members were working on?

- Yes.

How did you communicate with other team members during the project (e.g. email, walk to cubicle, phone call, IM)

- I preferred whenever the member was here, face to face contact. But because <project manager> is in Blacksburg and <usability engineer> also is not there, we also did use IM and emails.

Did you ever have a disagreement with another team member(s)? How was this resolved?

Do you feel you received the support you needed from other team members to carry out your role in the project?

How well overall do you feel the team collaborated? With respect to other projects you've been on?

Usability

How well did the usability engineer work within the team?

- I mean at first there was a struggle of how is this going to work? Because this was also as I understand <usability engineer>'s first project. Also the team was not used to working with a usability engineer. There was a bit of confusion about the role of the usability engineer. After that got resolved, it worked pretty well I think.
- Also issue with parttime UE mentioned above.
- One thing I noticed with <web technology> development. There were two different tools. One, visual studio to code the project, and expression blend for designing the UI. In an ideal scenario, instead of powerpoint mockups, if the UI could design the ZUI in expression blend studio, that would have worked amazingly. That would have been a prototype that could have been turned into a product. And issues of interactivity would come up in design time.

Is there value in having a separate usability engineer role on the team? What did the usability engineer contribute to the team?

- Definitely. If there was not someone 100% thinking about the usability of the application, there was no way we would have completed the project in this time frame with these results.
-

Did you understand the sorts of things that the usability engineer did and why?

- Aside from personal design skills, I think she utilized scientific results and tests to determine how the design should work. Like she did user tests and stuff like that and depending on that, she would change things and those things really made sense. And that was also one thing the team really liked, and we started demanding more of that towards the end. If something was to be changed, we were like why do we want to change this. And if the rationale and user reactions could be laid out, everyone felt like that was the right thing to do. And some things we discuss things needlessly for a long time. When we have rationale laid out and user data gathered, that just...everyone accepts it.

What was your relationship with the usability engineer?

- At first, our communication happened through the mockups. As the interactivity issues started to come up, several times we were able to do this—we would sit together, I would show her options and change things on the fly. We could just iterate through things together. We could only do that several times because our schedules didn't overlap together.

How well did you communicate with the usability engineer?

How important were usability evaluations in the conduct of the project?

How does this compare to other projects you've been involved with?

What benefits/problems were encountered by incrementally developing the user interface? How does this compare with past projects?

- Let me start with a benefit, with the new .Net UI stuff, there's the opportunity to just put out the bare bones of the UI in there. It could look really bare bones, but because of the way it works, you can improve it without just changing things. That kind of thing fits into the agile process because we didn't have to develop throwaway prototypes. We worked on the same code base to bring that up to the final look.
- However, when a workflow change would be introduced, we ran into issues with the underlying code because the code was intended to work with the previous workflow and when you change the workflow it became really challenging (should I patch the code or rewrite it?) That became frustrating because the schedule was so tight and the scope kept increasing. And when we ran into these reliability issues we had to refactor anyway. I don't think this can ever be avoided. You give a programmer some requirements and the programmer will code to those requirements and not some undefined future requirements that come up.

Did incremental develop affect the overall quality of the user interface? Compared to designing the UI up-front?

What was the role of the usability tests with regard to the overall product? What did they contribute if anything? How does this compare to past projects?

See above

Did you believe that usability design decisions were made with respect to high level goals of the project? Were any changes not aligned with high level goals?

- Yeah. As we worked. As the team got used to working with a usability engineer. They started demanding rationale, user reactions. All the decisions out of the usability engineer became decisions that only affected the goal.
- Two weeks before release <usability engineer> changed the background color and look and feel of nav buttons and it just looked good and I just implemented it. So we did take a few liberties to have some fun.

-

-

Were there conflicts between team members with respect to which usability changes that need to be made? How were these resolved? (tradeoffs?)

What was the effect of having usability and development working in parallel? (generally having the usability engineer work 1 iteration ahead of the developer) What were the benefits? What were the problems?

- It's great that the developer doesn't have to worry about how the UI is going to look like. When you are developing from scratch and you have time constraints, and you have a design, then you can do it efficiently. If you also have to worry about what it looks like, it really takes forever.

Were there conflicts between the UI design and development constraints? How were these resolved?

- There were conflicts mostly regarding resource constraints, technical difficulties and possibilities. They were resolved by discussions and estimates the developer would give for a certain feature at the meetings.

Information sharing/Project management

How often did you refer to usability documentation on the portal? How was it used?

- I referred to the customer artifacts a lot. And the mockups a lot. But, I did not refer to anything else. I think that most of the documentation that usability engineer did was a waste of her time. Because no one in the team read them and she could have just told us her thoughts and that would have been enough. And it also went against what we were trying to do with agile and that's wrong.

Goals?

Scenarios?

Mockups?

Claims/design tradeoffs?

Iteration plan?

How did you share information with the team?

- I also had a document up there on the portal. But that quickly became useless so at first it was just verbal communication during the team meeting. But when the work item system got implemented, a lot of the information was shared through work items.

Did you encounter any problems sharing information with the team or getting information related to the project?

- Yeah. I mean especially at the beginning, I couldn't get finalized mockups for the UE. I

couldn't get work items from the team lead for new features. I also didn't get any work items from the UE. And also a lot of the bugs that were written at first contained almost no useful information (some written by team lead and some from QA)

How aware were you of development progress throughout the project? Were you ever surprised by changes made to the system that you didn't expect?

- Yes I was very aware of it. I was pretty much in control of everything that was being done. So I was never in surprise mode.

How did you use Team System if at all during the project? Did you notice any problems with how it was being used?

- I used it to check in code. Deal with bug reports and change requests. If we were discussing anything, we would track the issue with a work item. That's something we should've done from day 0. You can relate work items to checked in source code so it's nice to do that.

How important were the different types of meetings held during the project (daily, feature team, review meetings with customer)?

- The customer one was the most useful. But then again because of our lack of effectively tracking requirements and such, we had so many moot discussions about things (the best example is the keyboard—it's just stupid for how long we discussed it because the time of implementation was shorter than the discussion really)
- About half the time, daily meetings were pretty useless b/c we weren't effectively tracking issues and writing. A lot of times we discussed the same thing and that was just irritating. (sometimes he says need to write things down, and sometimes we don't need to)

How well did the team stay on schedule during the project?

How would you compare project velocity to other projects you have been involved in?

- This was really, really fast. Very rapid. The amount of changes we were able to make was staggering. If you were to pull up the initial mockups and compare to how the application looks/works now, I would've laughed at you that it wouldn't be possible.

What things were done to address problems with project velocity?

How often did requirements change throughout the course of the project? How were they handled?

How does this compare to past projects you've been involved in?

Others

What are your thoughts on the development process used for this project?

- I think the whole process and tools and technology were really compatible with each other. It's just that the team was inexperienced with the process and the tools and the technology.

How does it compare to other projects you've been on?

What things would you have done differently?

Any other problems/issues you encountered?

Developer 2

General questions

What was your role in the project? What did you do?

- I guess the best way to look at it would be 'consultant'. Depending on the amount of time availability on <developer>' side I would implement stuff if needed. I made sure things were running fine in terms of design and implementation and scheduling.
- Average about 15 hours a week.
- I was developing <handheld> and <handheld 2>.

Project goals

What were the key design goals for the system?

- The key design goal was a scalable way of deploying the application or the service. Well structured, easy to understand workflow and interface that users at <customer company> would be able to work with. A good, technical implementation to interface data from <BMS>. From a performance standpoint, we needed to be very quick in rendering data to the user and not having too much lag and it had to be real time.

Do you believe everyone had a clear understanding of the goals? Were everyone committed to those goals?

- I think there was a little bit of confusion early on where I think the technical personnel were not willing to go ahead and define the design goals and requirements. They were expecting them from the customer. There were misunderstanding in terms of what the design needed to be and what it needed to address.
- I think it was resolved by proactive/aggressive stance by certain team members to make sure the project was designed in a certain way. Within development and product management.

Were there goals that conflicted with one another? How were they resolved?

- The conflicts were basically resolved by someone coming to say this is how it was going to be done. So it was almost brute force to make sure the team follows a certain direction. (Mainly product management—e.g. it has to be a web application rather than a smart client).

Were these goals used to guide design decisions in the project?

- Yes. Scalability (lot of people can start using the software very quickly—there's not much installation—many people can come and use it without having to have installers) was the biggest driver. Now there's also one other circumstantial/environmental parameter that affects scalability which is lot of people think it should scale to smart clients as well. The fact is that the IT departments of big corporations are extremely slow and bureaucratic in implementing/pushing new updates to machines so that clients do get the latest version of a software as opposed to a web client where you push it to the server and the user immediately has access to it.

Do you think the team met the project goals? Why do you believe that it has or hasn't?

- I think they did. I think they met them because as we discussed earlier we needed the application to be easy to use, which I think the application is. I think based on some of the user tests and some of the demos... The customer himself is happy so I guess that's the greatest validation.
- From a design and technical standpoint, it works.

Did the team always have a common understanding of what things needed to be worked on and why?

- From the standpoint of UI, I think they did. From a backend perspective, there were a lot of things that were missed. Over time, we discovered a lot of things that should have been different. But where we're most comfortable is that the UI is great. But it is new technology to the person who is using it.

Customer relationship

How important was it to be in contact with the customer/client during this project?

- I think it could have been less or maybe certain phases of UI design—great involvement, great discussion. But there are certain phases where you don't really need the customer to jump in and tell you what's going on. You need to reach certain tangible areas that you can show. Some software engineering tasks take time—when it's done you can go and involve the customer.
- Having a customer on hand, I think, got this mindset going with many of the people in the project, where we can ask them a question so we don't need to understand it completely. In other projects, you have a good written down version of what the customer is saying. There was not a good way of capturing what the customer was saying. All of these were not taken down and were getting lost. If it was preserved, I think if that had been done, it would not have been needed as much.
- As you're talking with the customer, the customer sits there and thinks about it, it's a scenario that's been discussed before. It's an environment, where there's no reference—we discussed

this before—this is how we captured it. Put on the screen, this is where it was, this is what was said. It was not happening. So you were always posing questions to customer and said it again and again. Instead of having a repository of what was said.

What benefits were derived from interactions with the customer/client?

- The best benefit that I see is I know when this guy gets the application, there's nothing hidden from him. There's no--He can't come and say you know what I want the application to be able to do this particular work flow in this way. There is lot of validation that is happening because of constant customer involvement. Secondly, I would say, it created an understanding of the project solution that you usually don't get to see. In most projects here at Meridium, you don't have customer involvement, so there isn't as much clarity (in terms of what needs to be done and why). When customer is there with you, you're getting an answer right there. You know it's right—at least you can trust it.

How does the amount/quality of contact compare to other projects you've been on?

See above

Did you feel that there should have been more/less customer contact? What should have been different with respect to this?

See above

Team Collaboration

How well did team members collaborate with one another? Did you notice certain problems related to how well one person worked with another? What caused these problems?

- I think it wasn't as glued as I usually had it in my other projects. There was a lot of different opinions coming in in terms of how things should be done. It created a lot of uncomfortable situations. It was a new way of doing things, so people don't want to change how they do things.
- And we had a project manager who wasn't experienced with managing developing projects here. (Scheduling issues).
- I think what we could have done is one major technical lead would have solved a lot of problems but didn't happen because of resource problems.
- The UE and SE that were on this project did an excellent job of understanding what the user wanted and how things should be done. From a UI standpoint it was a success.
- Also when you're in a customer meeting, having that many people in a room—that's an intense task. I honestly feel that product manager with technical lead sitting in one room together and communicating it with the rest of the team is probably a very good way of doing things. Having everyone there to ask questions is nervewracking for the customer and created more confusion.

What sorts of problems were encountered and what caused them?

- For me, the biggest collaboration issue was the allocation of my time in the project. The

allocation of resources in the project. And just having technical discussions. They were not very well executed in terms of communicating to everybody where we stand technically. Where are the deadlines. What do we need to do to accelerate ourselves. I just feel that there were a couple of people who knew what were going on. And the rest of the people did not have any clue what was going on in the project. You need a Proj. Mgr who understands these things. We did not have a code review in the entire project. How was a design review not done in the entire project. You need to have a design review.

- If there was any resolution it happened through escalation. It would not happen, people would get freaked out and things would get forced.

Did you ever feel that certain team members were not involved enough or that they were overstepping bounds?

See above

Were you typically aware of what other team members were working on?

- Um, yeah. Usually I had an idea about it. I think QA was a little lost. I think QA needed to be a little bit more involved in understanding the goals of the project in terms of the functional goals. What the project was meant to do. We discovered a lot of things now that should have been discovered in phase 1 in terms of downloading a work order, updating a work order or operation.

How did you communicate with other team members during the project (e.g. email, walk to cubicle, phone call, IM)

- When I was in <loc>, I communicated over phone. When here, lot of meetings. Had standups with <developer> to understand what was going on. My main goal was to be able to understand that we were hitting time, and the thing works.

Did you ever have a disagreement with another team member(s)? How was this resolved?

- I think there were disagreements from the standpoint of priorities and in terms of what things needed to be done. I was forceful in terms of design reviews. You sit down together and you give reasons for why things need to happen. The surprising thing was it was not done by other people. I should not have been required to go in and say this is how things should have been done.

Do you feel you received the support you needed from other team members to carry out your role in the project?

- I think I didn't get support from <project manager> in terms of... I didn't think he communicated with me effectively where we were falling in our timeline and technical goals. For a certain time, I thought maybe everything is fine. As phase 2 started and I got more involved, I thought...
- I honest think there was a lot of effort given to how the UI looks, and it looks good, but we've refactored so much of the code lately, it's just amazing. I couldn't understand why it was done in such a shabby matter. <project manager> said maybe because of the process we weren't able to go back and refactor stuff. We never really went back and looked how things were coded to see how they support current developments. And that could not have happened

for various reasons (limited people and time resources).

How well overall do you feel the team collaborated? With respect to other projects you've been on?

- I don't think we collaborated really well. I think there were a couple people completed outshined the project in terms of doing it how it was. They were 100% committed to the project. A lot of times people would approach me and say are we gonna make it? There wasn't enough trust in the leadership. That's a huge problem in terms of leadership. There was no clear direction in terms of how we're doing things, what are the problems, how can we address them? I think a lot of people just kept quiet. We missed out on good teamwork. I think a lot of this is people's experience with this project. I wasn't communicated with very well. I think this limited the scope of what we could've achieved.

Usability

How well did the usability engineer work within the team?

- I think she worked perfectly. She understood the requirements from the customer very well. She communicated with the customer very well. Of course she came up with a good design. From an underlying perspective, I think she did a whole lot of documentation. And I think this helped her understand what this is all about. Which might have affected getting good UI.
- I don't know how much it is being a UE versus being a person who is dedicated to draw up a UI.

Is there value in having a separate usability engineer role on the team? What did the usability engineer contribute to the team?

- I think it is necessary to have a usability engineer. But, let me rephrase this. I think it's practical. But I think the underlying issue is there needs to be time dedicated to design a UI. Whether it's done by the UE or by a developer, it's more important that the time is set aside to do it.
-

Did you understand the sorts of things that the usability engineer did and why?

- I mean her, drawing up scenarios for example. I think it helped her to get an insight into the environment of the customer. The day in and day out that these people are in. Drawing up scenarios helped.
- Collecting artifacts (like how do they do these error codes) gives you a picture as to how do these people visualize what they're doing from an information standpoint. How is it that they're doing it now. It gives you a structure of how things work right now. Which usually we (Meridium) don't do.

What was your relationship with the usability engineer?

- I did not work with her as much as I would've liked. She would bring the information that she collected, and give insights in terms of what the application is about. She would talk in the meetings and give me an idea of what the customer is thinking.
- And when she showed the UI, it helped me understand how we should work to get it working. And question her in terms of what was her thinking behind it. Why the Ue did it a certain way so you can get sold in terms of what the UE was doing.

How well did you communicate with the usability engineer?

How important were usability evaluations in the conduct of the project?

- I don't know I wasn't involved in that. I know they were done so they got be confident about the UI. All software engineers kind of ask how the UI could be better when you know a user has gone through it, that's the end of the discussion. For phase 2, we've had these discussions. At the end of the day, there's been someone who's done an evaluation and told us how it should be done. And there's been so many instances where everyone has very different ideas about how things should be done. That's a great way of solving UI I think.

How does this compare to other projects you've been involved with?

What benefits/problems were encountered by incrementally developing the user interface? How does this compare with past projects?

- There's complete coherence when it's done all in one go. There is a long term view. Long term views always prevail over short term views because they take into consideration all that's coming. Short term views you don't have the ability to sit back and look at it as a whole because you're always trying to find the piece that would go and look into the one that you just developed.

Did incremental develop affect the overall quality of the user interface? Compared to designing the UI up-front?

- In this case, the features were very well forked out so that issue did not come about. I'm always of the consideration that you need a long term view of the design that you're doing. I like to know what features I'm implemented. Whenever you're designing a new system you have to understand at some level what is your framework. Then you start talking about we need some sort of platform where we're able to support different things. Automatically the entire scenario changes.
- Most of UI problems come because there's not enough consideration to what needs to be added later on. If you know what's coming up then you have a good idea.

What was the role of the usability tests with regard to the overall product? What did they contribute if anything? How does this compare to past projects?

- See above

Did you believe that usability design decisions were made with respect to high level goals of the project? Were any changes not aligned with high level goals?

- Yeah, I think so. I don't think there was any aspect that did not achieve high level goals.

Were there conflicts between team members with respect to which usability changes that need to be made? How were these resolved? (tradeoffs?)

- There was, but I think we had a discussion about it. I took myself out of the UI because I knew someone was handling it.

What was the effect of having usability and development working in parallel? (generally having the usability engineer work 1 iteration ahead of the developer) What were the benefits? What were the problems?

- I'd pass on this question. I don't think I had enough to do to answer those. From an early inception standpoint, when you're trying to put certain things together, you're trying to draw a picture of how things need to come together. I know people wanted to know how things would work at a high level. When <usability engineer> came out and presented the navigation panel, that was OK, we know this is gonna work.

Were there conflicts between the UI design and development constraints? How were these resolved?

Information sharing/Project management

How often did you refer to usability documentation on the portal? How was it used?

- I only looked at the screen mockups that <usability engineer> put up and I looked at the artifacts.

Goals?

Scenarios?

Mockups?

Claims/design tradeoffs?

Iteration plan?

How did you share information with the team?

- Discussions and chats.

Did you encounter any problems sharing information with the team or getting information related to the project?

How aware were you of development progress throughout the project? Were you ever surprised by

changes made to the system that you didn't expect?

- Very little, miniscule. And I completely detested that fact.

How did you use Team System if at all during the project? Did you notice any problems with how it was being used?

- I did but I wasn't managing the project so I didn't think my task was to go and look at these things. I needed someone to come and draw the bigger question for me. That wasn't done. And also the fact that sometimes you need to understand where the team was and you can go and help.

How important were the different types of meetings held during the project (daily, feature team, review meetings with customer)?

- Standups with <developer> (see above)
- What really turned me off is there was so much customer-centric focus. There was almost a time where I thought is the customer driving the project or are we driving the project. I think the customer role should be to give feedback on what we develop. To wait for small decisions from customers... all we're talking about is what the customer is thinking. That doesn't give due time to the other aspects of a project like resources.

How well did the team stay on schedule during the project?

- We were making compromises at the end of it, what are we doing and what w're not doing it. We met phase I goals but in terms of how those were compromised I'm not sure. From a UI standpoint it was done—so from that standpoint it was great.

How would you compare project velocity to other projects you have been involved in?

- I think there were a few people that were very vested into it and driving hard. Unfortunately in a professional project it's a show of many people and many people are trying to run in different directions your velocity comes close to 0. Even though we got through, coherently we did not do a very good job.

What things were done to address problems with project velocity?

How often did requirements change throughout the course of the project? How were they handled?

How does this compare to past projects you've been involved in?

- We seemed to be on track. There was little cases where things changed but there wasn't much change. I was actually quite impressed by that. From a requirements standpoint I think we did very well. I think customer involvement helped that. I also think that PM...the PM did not need to answer as many questions.

Others

What are your thoughts on the development process used for this project?

- I think in retrospective if I were to do this thing again, I would try to add more structure in

terms of the way the show is conducted. I would try to get more buy in for more people. I don't think many people bought into the process. I would very, very focused into what I need from the customer—how much I want him involved. I would not be waiting for the customer to give me a go ahead.

- I had things like I want to give the customer what he wants, but I think that's a very idealistic statement. The customer will never stop saying what he wants. You need to be prudent in defining how far you're ready to go in terms of getting changes done to the project.

How does it compare to other projects you've been on?

What things would you have done differently?

Any other problems/issues you encountered?

Usability Engineer

General questions

What was your role in the project? What did you do?

- I was the usability engineer.
- I developed UI mockups based on info I collected from the problem domain, the end user and the customer and also my teammates. I worked closely with the developer to coordinate changes made to the UI. I evaluated the UI prototypes and working versions.
- I did 8 hrs, 3 times a week.

Project goals

What were the key design goals for the system?

- I think it was time efficiency, user acceptance and high quality data.

Do you believe everyone had a clear understanding of the goals? Were everyone committed to those goals?

- No. I think I presented them once in a meeting, and I sent them in a email and I had them posted to the team portal... but some people weren't at the meeting and even if they were maybe they didn't think it was important. I doubt everyone read the email and understood why those goals were important. And I have the suspicion that not that many people looked at the usability documents. I think the reasons for those things was that the teammates did not necessarily understand what my process was and how those parts of the process were important to them. In which case they didn't understand and wouldn't make an effort to learn them. Although I have no evidence of people not knowing the goals.
- I think each person had different goals for the project. I think <product manager>'s goal for

the proj. was more to develop a professional relationship with <customer company>. To create a product that sold be sold to other companies and to spend moderate time and effort to get a working marketable product. I know usability was definitely a secondary goal to that. I think <project manager> was interested in securing respect and experience and proving himself as a manager. And perhaps one of the ways he did that was to take advice from <product manager> and look up to people that had been doing this for a while. <developer>, this was his chance to show off his technical prowess. This was his first time being lead developer on the product so it was important to him that he met deadlines within reason and that he made something that worked and had some flashy aspects to it. And I'm guessing that people like <documenter> and <qa consultant> were just trying to minimize the amount of work they had to do. I think they were worried about the agile process and be a lot of work. Just trying to do their own thing without keeping up with what everyone else was doing. I think <qa person> was new to th company and just wanted to show that he was a quality worker. And he put a lot of effort into the team and got really involved. I think his goal was really to make a high quality product. Not necessarily caring about money.

Were there goals that conflicted with one another? How were they resolved?

- Sometimes yeah. Usually you see when they conflict. There were issues around time and <developer>. I mean he wanted to feel like he could accomplish the work he was given and it was something he could manage in the time he was given. I think issues came up with <project manager> when –I don't think he really listened to the team about some management ideas because I think he was trying to assert his authority or feel that he was in control of the situation. I think there was issues with <qa manager> and <developer> and how he was scheduling and naming things and writing bugs. I think there wer issues with <product manager> where we were focused a quality <customer company> product versus making a generalizable across the industry product. And also issues when the usability team wanted to do something would compromise meridium's ability to create a professional relationship. I didn't really see any issues with <qa person> or <documenter> or <qa consultant>. I think that <documenter> and <qa consultant> were withdrawn from the group-at a distance. <qa person> was very flexible really wanting to give and he wasn't as rigid as other people.

Were these goals used to guide design decisions in the project?

Do you think the team met the project goals? Why do you believe that it has or hasn't?

- I think we did. I think it could have been better but I think we under the situation came up with a decent product based on those three things.

Why did the project still succeed

- There is the fact that I had the gaosl internalized and fought for every day that I was there. On the other hand, it was not that the other team mates did not care about these things also. To the extent that it didn't conflict with their other goals, I think they also supported these goals subconsciously. I think that when important decisions were being made (with respect to the 3 goals) I'd recognize many of them and try to fight for them.

Did the team always have a common understanding of what things needed to be worked on and why?

- See above

Customer relationship

How important was it to be in contact with the customer/client during this project?

- I think it was critical. Especially toward the beginning because we had almost no idea what the current situation was and the problem is we were trying to solve. I think the site visit helped to really narrow what we were being asked to do. But even after that there were lots of questions. Big things that were answered in the first 3-5 meetings with <customer representative>. If we had not had that interaction, the assumptions we had made – we would have ended up with a product that looked very different and suited needs much more poorly. Or even simply didn't work. Even technical things got worked out through correspondence with <customer representative> (I think it was the stuff about the activity type and instead of showing the mechanic the activity code we wanted to ask them easier questions to come up with the activity code in the background. We ended up asking <customer representative> how this could be done.)

What benefits were derived from interactions with the customer/client?

How does the amount/quality of contact compare to other projects you've been on?

- One important thing to come up was that I wasn't in a position to talk to customers. I think what happens in that model I could potentially be making decisions that don't reflect the customer need cause I did some things with the UI and we didn't really have any idea who the customer was at all. In fact, I don't think the product I was on at IBM had a specific customer. To my knowledge no one went out and interviewed the customer. I think they were designing for themselves. The usability on that product was really poor. I was a developer and I had trouble understanding how to do certain things in the interface because of the labeling, etc. And that's what actually got me interested in usability. There was a human factors person that was split between several projects. She had a much smaller role in the product. The developers would do the screens and then the human factors would make small comments.

Did you feel that there should have been more/less customer contact? What should have been different with respect to this?

- I think it should have been more (contact) and different. Mostly I say more because we didn't really get to talk to the mechanics. We didn't get to immerse ourselves in their environment and learn the little things that really add up. Like their attitudes towards technology or anything from that to where they like to hang out and spend time. We should have observed the mechanics, had interviews with several of them. And had 2 or 3 more physical visits to the site where we demonstrated the prototype. In terms of meeting with <customer representative>, I feel like if the project had been better managed, then it would have been useful to meet with <customer representative> more often and he wouldn't have minded. If we had a very clear agenda when we met with him, and a very well, professionally presented demo and points of feedback. But I always felt like they were really sloppy and they do make

the company look bad when they're sloppy that way.

Team Collaboration

How well did team members collaborate with one another? Did you notice certain problems related to how well one person worked with another? What caused these problems?

- Well, I mean it ranged from very good with <qa person> who was so kind to help me out with my studies and doing some wonderful things as he was reviewing the UI providing feedback on what he thought about the interface. He was also acting as the critical user which was helpful, showed he cared.
- I think I had a lot of tension with <developer>. Cause we each had different visions of how the interface should work and how much effort we should put into making the interface that way. So oftentimes I would deliver a usability mockup to him and he would usually develop most of the features I asked for and then others he would develop his own instantiation of the feature because he thought it was better his way. Or he would spend a lot of time on things I thought were not important. It was really hard for me to rework the interface. It was really hard to get him to go back and fix the interface that he did his way. So I had to go back and redo the UI to work with how he did things.
- I felt like <project manager> didn't trust me with some things like the questionnaire I was going to give to the mechanics or with the interface at points. And other things like he would spend a lot more time on things I didn't think were important. So I felt tension between wanting to play along with <project manager> versus getting real work done. He's sit down with me 2 hrs to plan out what I would do for 2 weeks but he wouldn't let me just sit down for .5 hr and explain it to him and then have him go and put it in the plan his way. That time, the UE does not need to sit with the manager to do his job.
- And then <product manager>. Biggest communication breakdowns happened with <product manager>. Who just wouldn't come to meetings and wouldn't respond to emails and would make decisions that greatly affected me without tell me and I would be really surprised. Like the decision not to have interviews with stakeholders at <customer company>. Or the decision not to use my design for phase 2, but to do something of much lower quality.

What sorts of problems were encountered and what caused them?

- I think it's mismatched goals. Everyone has different goals/different priorities. I think it's seniority and respect for position. So <product manager> has been there for a long time, I was very new. Or the software engineer is valued more than the UE which is valued more than documentation so there's definitely those things. I think sometimes, um, understanding of expectation—what I expect from a manager—you are good at being a manager so you don't waste my time. Or my expectation that I'm not going to be pulled into a meeting where everybody is discussing how the UI should be. So expectation of roles. Maybe some of it is just personality and how you like to deal with issues. I think <product manager> just sort of, he guards his information and decides who he gives certain bits of information to—it gives him power. <developer> would just do things without asking for permission and then afterwards defend his actions ruthlessly.

Were any of these issues addressed?

- I think one block that prevented these things from being addressed was openness. We're in a professional environment. I think it makes it difficult to discuss things like (those above). It's not easy to hold those conversations at least for me. I think the process helped a little bit with these issues in that it formalized certain key things like the usability engineer first makes a mockup then the coder codes it so that gave me a power—the UI person is pushing the UI down. The meeting schedule was good to an extent at least I could detect when <developer> had made changes to the UI and then those could be addressed quickly instead of. I think it broke down in matters where <product manager> just didn't show up to meetings. And when <project manager> started meeting at Blacksburg. Having remote meetings—I don't think that's a good idea. Another thing is the process formalized meetings with the customer which I think we would have had far fewer meetings if we didn't have the process. I guess what the process does is despite the varying goals of the teammates it does impose these teamwide goals if that's what we want to call them. (e.g. it's important the Ue is designing the interface.) In that sense it's the goal that helps us travel together.

Did you ever feel that certain team members were not involved enough or that they were overstepping bounds?

- See above

Were you typically aware of what other team members were working on?

- To different degrees. <developer> was right across the hall which really helped cause he's just go hey <usability engineer> come here or I'd go hey <developer> come look. We'd get a lot of instant feedback. I had to know what state the UI was in so that forced us to hooking into what each other was doing (because I ended up having to evaluate the UI every week)
- <qa person> and I, it really was a different story. We weren't physically that close but we connected a lot through email cause he took the time to send me great emails and that ended up with me writing a long detailed email to him or me going over there and having a discussion with him. So his motivation to do a great job with testing—which means he had to keep track of how the interface was being developed—had him get stuff from me.
- The developer and *had a* bidirectional relationship.
- The tester and I is more of a unidirectional arrow. He needed stuff from me.
- And I was sort of aware of what <project manager> was doing because he had to be at every meeting, he was the manager.
- But <product manager> and <qa consultant> and <documenter> were pretty much off the map. I had very little sense of what they were doing.

How did you communicate with other team members during the project (e.g. email, walk to cubicle, phone call, IM)

- See above

Did you ever have a disagreement with another team member(s)? How was this resolved?

- See questions above (<developer>, <project manager>-not open communication)

Do you feel you received the support you needed from other team members to carry out your role in the project?

- Eventually yes. The beginning, I think <project manager> was mostly on my side throughout he just this very blanket idea that usability is good. Specific instances he was not very supportive (especially that one instance where he said he didn't like my questionnaire). He usually made sure to give me a spot in the meeting. The developer got a lot of time but then I got time too. I think that's important to note.
- With respect to difference between the beginning and the end. I didn't really have the support of the team until I demonstrated to them that I had the ability to do quality UI work. I think that happened in the meeting where everyone was trying to design the UI, bombarding the UI design. Maybe that was a good conversation to have because they saw that I had reasons to do the things I did. And I got the support of <developer 2>. The reason I got respect in that meeting—I mean everyone feels like a UI designer to some extent—and I don't think they saw until then you need someone who considers how it all flows together and that I was that person and I wasn't going to let them down.

How well overall do you feel the team collaborated? With respect to other projects you've been on?

- I think it was better—even though we had all those problems with people not showing up to every meeting. For the most part I got to see everybody on a decently regular basis. That didn't always happen on my other teams. I'm trying to think back to IBM. We did have weekly team meetings but there was a lot less of a discussion. Seemed like much more of a 'you're gonna do this' rather than a discussion of what should be done next and what timeframe. Most of us had a stake in what was being discussed.

Usability

How well did the usability engineer work within the team?

- I've gotten a lot of feedback, that is, made me feel like I was too pushy. So I feel like maybe I wasn't enough of a 'team actor'. Maybe I was too selfish and not understanding of other people's needs. But I do think that I took actions that made important differences on the product.

Is there value in having a separate usability engineer role on the team? What did the usability engineer contribute to the team?

- Yes, see above.

Did you understand the sorts of things that the usability engineer did and why? (Did you understand what others did and why?)

What was your relationship with the usability engineer?

(As the UE what was your relationship to other members of the team)

- See above.

How well did you communicate with the usability engineer?

(How well did you communicate with others on the team?)

How important were usability evaluations in the conduct of the project?

How does this compare to other projects you've been involved with?

What benefits/problems were encountered by incrementally developing the user interface? How does this compare with past projects?

- I mean let's start with problems. Problems I think were that I had a lot to do up front. You can't just say I'm going to design half the system. You do have to think at least abstractly about the other parts of the system. For instance, the theme, the method of navigation. The method of navigation is obviously impacted by how much content that's going to be developed I had to think about them. And what set of actions are they going to want to do because for these actions I'm going to have to create a small set of functionality. And I don't want them to lean a billion different looks and feels. And will this set of buttons or widgets support them. You have to think about broader issues. And it definitely made me worry that I had to make those decisions up front without considering everything.
- But the good stuff, it's good because it would've taken me much longer to do all of it in one sitdown and development would have been paused for that time. Another thing is that my first vision of what the system needed to do was not correct. It was continually tweaked by the customer so the interface ended up changing anyway. I think one more positive/negative tradeoff that there is if you do all the development up front you might get a more consistent interface that doesn't necessarily fit the customer as well. If you do it incrementally, You'll get a decently consistent interface that fits the customer better.

Did incremental develop affect the overall quality of the user interface? Compared to designing the UI up-front?

- See above

What was the role of the usability tests with regard to the overall product? What did they contribute if anything? How does this compare to past projects?

- I think they were like a reality check. As a designer you live in a dreamland, world that you get to simulate in your head but as soon as you test that prototype in the real world, you see the things that were missing from your simulation that you hadn't considered. I think they also served as a way to help me engage with my teammates. Have discussions and negotiations with teammates about what aspects were important. The results of the study create a neutral source of evidence that can help engage group discussion and negotiation.

Did you believe that usability design decisions were made with respect to high level goals of the project? Were any changes not aligned with high level goals?

- That was always part of the decision. And it really helped to have them written down. It really helped to have them. It became more of a concrete –something concrete for me to focus on as I made my decisions. But there were other things I thought in the back of my

head that affected what I thought was an acceptable design decision. But at the end of the day, those three had the biggest influence. And they were prioritized which helped because some decisions presented a tradeoff between (the goals).

How were claims used in the design of the product?

- I have to admit. Claims became more of an afterthought. After I designed the interface with respect to the goals—I'd write the claims). Claims became a way for me to unpack why something is good enough to make it into design.
- When it came time to testing, I think it influenced the types of things I tested. I think I had designed a study before you should be using the claims to design the study. And I had to go back and redesign the study and I realized that I wasn't looking for all the right things in the study. They helped me keep focus and overall they helped me get the evidence I needed to open up discussions with teammates about the interface.

Were there conflicts between team members with respect to which usability changes that need to be made? How were these resolved? (tradeoffs?)

- One stands out to me with respect to <project manager>. He said that he just didn't like some feature. And I said I think it's important. And I said there's a real usability reason for this so let's try it my way first and then we'll test it and try it your way if it doesn't work.
- Sometimes those got resolved through the team discussion. I hated the summary page that linked back but the rest of the team liked it and didn't see a problem with it and my usability studies no one really cared that the people I tested it on didn't really know how to use it. I fought a few days for changing that and over and over the team said I like it the way it is so it just stayed.
- <qa manager> said when you touch a button you shouldn't touch 'next', but I said I wanted it to be consistent. but then the user study showed we should just do that so I just changed it.
- I did the interview with the expert touch screener and finally there's lots of things that played into it. There were other issues (time to complete). <developer> said it wouldn't take that much time to do it and let's just do it. He helped me push it over the edge.
- I know one of the things that happened was the selection lists were too small for fingers and then the change got made. So some things were just that simple.

What was the effect of having usability and development working in parallel? (generally having the usability engineer work 1 iteration ahead of the developer) What were the benefits? What were the problems?

- I think the benefit is that we don't feel like we're wasting time. That everyone is continually engaged in the project moving forward on some bit. I think the drawback is that I felt really rushed to get out the next version of the UI. Maybe that was an effect of the compactness of our iterations. I guess one issue is <developer> would be developing a mockup that I just gave him, I'd be working on a next iteration of the mockup and all of a sudden I'd realize that <developer> developed something different than what I prescribed. And it depends on when I discovered it. (I'd have to redo the mockups or if I'm doing user tests I'd not test that bit anymore cause it didn't exist anymore).
- I think when he's developing while I'm working on the next we're not yet sure if he's going to be able to get that chunk done so I'd work based on the assumption that he will be able to. I felt like at times I developed new ideas for features that could never be developed cause

features kept getting pushed to later iterations because they weren't done on time.

Were there conflicts between the UI design and development constraints? How were these resolved?

- See above

Information sharing/Project management

How often did you refer to usability documentation on the portal? How was it used?

- Early on, I used the docs created by <product manager> seeing as how I couldn't talk to him much. So I could get an idea of who I am designing for. At one point I was looking at the iteratin plans but then I think that those were a really poor representation—really confusing. I think <project manager> did a poor job of making those accessible so I just stopped looking at it. TO my knowledge other people weren't really posting stuff.

Goals?

Scenarios?

Mockups?

Claims/design tradeoffs?

Iteration plan?

How did you share information with the team?

- I had 3 levels of documents. The lowest level was all the nitpicky stuff I'd use to show what I'm currently working on. Above that were quick summaries of lower level documents that would be easy for teammates to read. And then there was one root node document that had links to the other ones. I should've done a better job of sending the link out to the team when I changed something big in it. With so few iterations it took a long time to put that structure up and get a routine going. Honestly I don't think anyone looked at it. If all the team members were dedicated to the project I think they would've done a better job of reading thorough each others documents.
- I relied more on meetings. Much easier to share info when everyone's in the same room together. Especially because that's what their formal purpose is. Plus you're face to face. Plus you need to share information. Why waste this time if we're required to be there. Although there were anomalies such as when <qa person> wrote those emails.

Did you encounter any problems sharing information with the team or getting information related to the project?

- See above- problems with <product manager>.
- See CI (where I had to call <developer>)
- I don't know.

How aware were you of development progress throughout the project? Were you ever surprised by changes made to the system that you didn't expect?

Yes, see above.

How did you use Team System if at all during the project? Did you notice any problems with how it was being used?

- I tried to use it to share my documents with people but I don't think they used it. I looked at it to see if anyone had looked at my documents.

How important were the different types of meetings held during the project (daily, feature team, review meetings with customer)?

- I think those meetings are good. But I feel like our meetings lacked focus so the benefit decayed with that. I think <project manager> could have done a better job of focusing feature team meetings so we could make progress. It seems like we wasted a lot of time. -There was no structure to the discussion.
- User meetings went alright—they could've been more professional, more planned out. I think before that meeting—we should've prepared somehow. Or maybe you just have an agenda setup before the meeting.
- The morning meetings usually got taken over by the developer or by talking about something that wasn't time effective.

How well did the team stay on schedule during the project?

- I was surprised that we stayed on decent schedule. I think we were only like a week late. But I think that we were, maybe our developer was working too hard. Maybe. I think that the last iteration was actually supposed to be minor weeks, we were working on kinda big stuff. I feel like I was pushed too hard to use usability evaluations. It's hard to do one of those every week when I'm only there 3 days a week. I felt like we could've been better at staying 'on track' if we had more structured feature team meetings. I mean that those meetings are where what features are up there, which are of highest priority and which will be done on time. And I never felt like there was a clear sense of. Something that we definitely got a little better at towards the end, but...

How would you compare project velocity to other projects you have been involved in?

- I've never worked on an agile team before. I felt like the work was a lot more evenly spaced during this project. We were making better use of people's time. It was more about a 'steady speed of work'. At IBM, there'd be 2 weeks where we'd do like nothing, just fixing little bugs. Then we'd have intense coding for a week or two where no one developed unit tests or documentation. It feels like there was a more steady progression.

What things were done to address problems with project velocity?

How often did requirements change throughout the course of the project? How were they handled?

How does this compare to past projects you've been involved in?

- Requirements are hard to define. I think that things changed often. Every week there'd be some small new thing or maybe a big new thing. Like whether or not there's going to be a

keyboard. That wasn't determined until the end. Those are pretty big requirements that are only addressed in the end.

Others

What are your thoughts on the development process used for this project?

I think this process is a step forward in making the usability engineer formally recognized as an important part of the team. But I think it tends to neglect the documentation team and even the testers. I think it's important to understand the formality of the process influences the social connections that take place there.

How does it compare to other projects you've been on?

What things would you have done differently?

Any other problems/issues you encountered?

Product Manager

General questions

What was your role in the project? What did you do?

- Product manager. Facilitating requirements (though it was done in team fashion for this project) , customer liason... to a degree an sme role.
- Not enough (time spent on this project). If I had to average it out, maybe 5-7 hrs.
- 6 other projects

Project goals

What were the key design goals for the system?

- Usability, Quick data entry, ease of use, concise—I think would be a good way of describing some of that. Obviously you got to get the accurate information—data and such.

Do you believe everyone had a clear understanding of the goals? Were everyone committed to those goals?

- Of the goals, I think we were fairly clear on what we were trying to do. The trip up to <customer company> helped to set those goals.
- I think so (everyone was committed).

Were there goals that conflicted with one another? How were they resolved?

- I mean, there was some conflict (challenging points) trying to develop this simple interface, concise interface yet dealing with some of the intricacies of <BMS> in the back end. Those things collided on more than one occasion.

Were these goals used to guide design decisions in the project?

- I'd say a majority, but not all. I say that b/c some decisions were made completely independent, without thinking of them. Such as the decision to interject meridium into the middle (a business driver). That came from outside of the team—without consideration for goals.

Do you think the team met the project goals? Why do you believe that it has or hasn't?

- I think so. Well, I mean the end product—easy to use, concise, quick whatever else. I think we achieved those things.

Did the team always have a common understanding of what things needed to be worked on and why?

- No. I can probably think of an example or two. In general, the team struggled for several reasons. 1) the team did not have <BMS> expertise on it—so it's hard to build an app that interfaces with another system when you don't know anything about it. 2) I think we also faced some challenges b/c the team was conflicted—by conflicted I mean the team was in an awkward position of trying to meet customer requirements as we were getting them from <customer representative>, but not having the full backing of the company so to speak—that would come with a true product development cycle. An example would be my involvement—I would've been much more involved (if this were a true project).. I would've gotten more attention from other areas (documentation is a good example). 3) there was always an attitude of let's just get it done for the pilot, vs. we didn't do what was right—we just got it done. (we haven't tested this out in the customer environment). If we were doing it from a true product standpoint we would've been testing more, we would've had more customer involvement.

Customer relationship

How important was it to be in contact with the customer/client during this project?

- It was important, very important. Critical to the success of it.

What benefits were derived from interactions with the customer/client?

- Probably the biggest benefits are avoiding problems, as you're talking about functionality—having that interaction with the customer you're learning things and can fix things before it's too far down the line.
- Understanding the integrating to system, understanding workflow, the environment—those types of things.

How does the amount/quality of contact compare to other projects you've been on?

- **This one, the quantity was on the higher end of many of the projects I've been involved**

in. Now the quality was probably in the middle. I don't think <customer representative> alone was sufficient. We needed <customer representative>, we needed <BMS> experts, I think we would've benefited from a couple other people—get a mechanic involved b/c he would've given his perspective and he's the true customer of this.

- **All of the interaction was remote access. We never had direct access other than that first trip. I think there was more we could've gotten if the attitudes were a little different. ...Relationship building which is important for these things as well.**

Did you feel that there should have been more/less customer contact? What should have been different with respect to this?

- **See above answer.**

Team Collaboration

How well did team members collaborate with one another? Did you notice certain problems related to how well one person worked with another? What caused these problems?

- Overall, team collaboration was good. There were friction points and challenges. Some things I would've changed... Overall, the team did share ideas, shared thoughts. I don't know if anyone held back—which is always a danger.

What sorts of problems were encountered and what caused them?

- I think there were a couple issues that surfaced several times. 1) From a development perspective—I think dev got put into an awkward situation on more than one situation b/c he was singled out—put on the spot—forced to make decisions right then and there. The example that comes to mind the team doing estimates of tickets in a team environment, when in my mind a developer should fully understand an issue to generate an estimate—instead of just asking right then and there on the spot what it should be.
- On a positive note, more so on this project, we let the Usability person drive things. Which I think is effective and we would benefit from doing more of. And if we can use that usability person to lead the dev. Cycle—in this case the usab. Does the prototype work and then shows it, and it gets handed off to development. I think that can be a very effective way of doing the work—esp. when you have a UI-focused project. I think the Usab. Did a good job of it, but she did it largely cause of her own drive and will, not b/c the team let her do it. If someone did not have the same strong outgoing personality it might not have gone that way.

Did you ever feel that certain team members were not involved enough or that they were overstepping bounds?

- I don't think I was involved enough. (see above). At times, I personally struggled with <project manager>—some of the struggles were personality based, some were management style based. And I'd guess that I'm not the only one that felt that way. And if my guess is correct, I think that had a negative impact on the team.

Were you typically aware of what other team members were working on?

- Yeah, I think so. Cause we had our weekly meetings, daily standups that everyone was on the

same page.

How did you communicate with other team members during the project (e.g. email, walk to cubicle, phone call, IM)

- In those meetings. Emails, we used phone calls when <usability engineer> was remote. And the web sharing stuff.

Did you ever have a disagreement with another team member(s)? How was this resolved?

- It depends on the type of disagreement. For feature/function related disagreements, the customer became the tie-breaker. I would like to think that <usability engineer>'s work would be the second in command—b/c of the high focus on usability. There's also some technical decisions that drove things, but to a lesser degree.

Do you feel you received the support you needed from other team members to carry out your role in the project?

- Yeah, I would say I did. Again, I felt we had a lack of <BMS>. There was not support there where there should have been.

How well overall do you feel the team collaborated? With respect to other projects you've been on?

-see above

Usability

How well did the usability engineer work within the team?

- Very well. I think she did a fabulous job—she knew what she was doing.
- As far as usability driving the team, I thought we did pretty good on that as well. The team had that in mind going in, that continued to be a focus throughout. I like what this team has started with this focus on usability, and letting usability be a leader in the development process. It works well with me b/c I'm a visual person.

Is there value in having a separate usability engineer role on the team? What did the usability engineer contribute to the team?

- Yeah, I do. (see above)

Did you understand the sorts of things that the usability engineer did and why?

- I think so. I participated in all of it to some degree b/c I wanted to see the whole process. We learned a lot through it.
- Just another side note: whenever I talk about the kiosk to customers, I always somewhere in my speech I'll weave in a usability story about how it was developed. How we timed people bet/ clicks, looked at their interactions. It always resonates with people. When you tell a customer/prospect that it eases their mind a bit. It also gives meridium some additional credibility that we know how to develop software.

What was your relationship with the usability engineer?

- I reviewed prototypes/mockups and made suggestions. I guess I provided her with requirements.

How well did you communicate with the usability engineer?

- I mean it goes back to the same thing I said before. I think it went well but it would've helped with more time. There was always a challenge between lack of my time and lack of <usability engineer> being here.

How important were usability evaluations in the conduct of the project?

- It proved valuable. We got value out of that. It caused us to make changes in our UI design.

How does this compare to other projects you've been involved with?

- There was much more focus on usability with this team. The other projects have not have dedicated embedded usability.

What benefits/problems were encountered by incrementally developing the user interface? How does this compare with past projects?

- The challenges of incrementally doing it, you do this piece without thinking about the next piece so then you have to go back and change it. I think that sums up the majority of the challenges. I guess you can also in the front end design the application—then there's a work process that breaks up what you're doing. An example—I don't think we originally thought about the popups. Oncscreen keyboard is similar to that. But I think if you're going to follow this process this is something you have to plan for up front.
- The benefits of doing the UI incrementally, you get to pass it over to development and they can begin working so you can work on the next piece. You're working ahead. The other thing I think it does, it gives you the ability to focus on a feature, which sometimes get lost so you can fully think about one little piece in silo and get it right before moving onto the next one.

Did incremental develop affect the overall quality of the user interface? Compared to designing the UI up-front?

- See above.

What was the role of the usability tests with regard to the overall product? What did they contribute if anything? How does this compare to past projects?

Did you believe that usability design decisions were made with respect to high level goals of the project? Were any changes not aligned with high level goals?

- Yeah, in this case definitely.

Were there conflicts between team members with respect to which usability changes that need to be made? How were these resolved? (tradeoffs?)

- Yeah, there were. Let me think of an example... I can't think of a good example. The keyboard, back to that—that was one of those things where from a project standpoint I was saying no b/c I didn't want to take additional time on the project to develop the keyboard. From a usability perspective <usability engineer> was saying it was great to have this. Ultimately <developer> said I can do this it doesn't take any time and we went ahead and did it.
- We were doing this project working with <customer company>, however we want to work it with other customers. The customer will say one thing so she will work on the ui based on feedback and I might want to open it up and make it broader so it applies to more customers. The activity type that we did was originally 3 buttons but that's specific to <customer company>. We went with a popup so we could apply to other customers. It was a tradeoff.

What was the effect of having usability and development working in parallel? (generally having the usability engineer work 1 iteration ahead of the developer) What were the benefits? What were the problems?

- I think in some ways it was easier on this project than many of the projects we do. b/c this project overall was small. If we were talking about developing a full module with a larger team then that's just gonna make it more difficult to develop software the way we did it on this small team. I think the same rules apply but we need to scale things accordingly.

Were there conflicts between the UI design and development constraints? How were these resolved?

- See above

Information sharing/Project management

How often did you refer to usability documentation on the portal? How was it used?

- Probably rarely. I'm trying to remember what's on it so if I can't remember I probably didn't refer to it very often.

Goals?

Scenarios?

Mockups?

Claims/design tradeoffs?

Iteration plan?

How did you share information with the team?

Meetings... (see above)

Did you encounter any problems sharing information with the team or getting information related to

the project?

- **Nothing specific. I can always make the gripe about email.**
- **There is a portal thing that needs to be worked out and that is documenting the team meetings. We had these meetings and we were not good at documenting what was discussed and the decisions we made and why. Ultimately we'll get back to these things and rehash them. So on that note we would've benefited from better documentation of our decisions. We had <usability engineer>'s documents which were current and accurate (except towards the end).—it 's the technical discussions (that caused problems)—especially when you're dealing with a customer and we get the information. It'd be nice to have something you can reflect on.**

How aware were you of development progress throughout the project? Were you ever surprised by changes made to the system that you didn't expect?

- I don't think so. (see above)

How did you use Team System if at all during the project? Did you notice any problems with how it was being used?

- Not in this project, no. (except in meetings where it was brought up)

How important were the different types of meetings held during the project (daily, feature team, review meetings with customer)?

- Personally I got more out of the daily meetings than the meetings with al. I didn't get much out of the feature team meetings. I mean we did go through tickets and stuff but... I just tended to get more out of the other two.

How well did the team stay on schedule during the project?

- I think we did OK.

How would you compare project velocity to other projects you have been involved in?

- The speed itself seemed to be a little faster. It was a smaller project and contained so I think it could.

What things were done to address problems with project velocity?

How often did requirements change throughout the course of the project? How were they handled?

How does this compare to past projects you've been involved in?

- I mean, we didn't document a whole lot up front in terms of requirements. So there was...it's hard to say. My guess would be there were more changes but I don't want to cast a negative light on that b/c some of those changes are by design (the way the team was working we expected those changes to be there).
- I think a good word would be 'nimble'. This approach allowed the team to be nimble which is

something we don't ways have in our other teams.

Others

What are your thoughts on the development process used for this project?

How does it compare to other projects you've been on?

- I enjoyed it. I'd like to see more of this usability work. I think there's benefit in it and I think the quality of the product that comes out of this is...the perceived quality in the customer's mind is better. (one facet of the quality is improved.)

What things would you have done differently?

Any other problems/issues you encountered?

Quality Assurance Lead

General questions

What was your role in the project? What did you do?

- That's a good question. I'm the QA mgr but I go to certain meetings because <qa person> was very new. Mentor.
- More closely involved with Web Client, Handheld, and the core framework stuff.
- For touchscreen I was middle of the road in terms of involvement.

Project goals

What were the key design goals for the system?

- Ease of use and connection with <BMS>. The purpose was to have an easy app so they could do away with manual interaction. The most important goal was to make it wasy for the end user.

Do you believe everyone had a clear understanding of the goals? Were everyone committed to those goals?

- I think for the most part, yes.

Were there goals that conflicted with one another? How were they resolved?

- Maybe in terms of performance. I know at times we struggled a little bit with perceived performance and what was happening in the backend and ease of use. I don't think it was a big deal.

Were these goals used to guide design decisions in the project?

- I think for the most part yes. I think that maybe it may have been useful if we had kept reminding everyone of that.

Do you think the team met the project goals? Why do you believe that it has or hasn't?

- I think so. I would like to be able to validate it with users but at this point we're not there yet. Based on what I've seen. Note that I didn't go to any of the customer meetings.

Did the team always have a common understanding of what things needed to be worked on and why?

- No. I don't know that I can give concrete examples, but it seems like there was a lot of conflict in, mostly, team lead and development. Conflict meaning I don't know if communication was as good as it could it been.

Customer relationship

How important was it to be in contact with the customer/client during this project?

- I don't know that I can comment on that. I can comment and say that it seems like when we needed an we could go to the customer and get them. IN that respect, it was good if not better because it was a direct customer. We tend to use our internal customers moreso than external customers. In this project we did well in that area.

What benefits were derived from interactions with the customer/client?

How does the amount/quality of contact compare to other projects you've been on?

Did you feel that there should have been more/less customer contact? What should have been different with respect to this?

Team Collaboration

How well did team members collaborate with one another? Did you notice certain problems related to how well one person worked with another? What caused these problems?

- I think that it was an interesting team because there was a lot of inexperience in the team. The team lead, developers, qa—I mean everybody was inexperienced. And then having a team lead be inexperienced made it more difficult. And a lot of different personalities as well. I think it got better towards in the end. There was lack of trust. Surprisingly successful for some of the communication , I don't' want to say breakdown but,. There were some problems.

What sorts of problems were encountered and what caused them?

Did you ever feel that certain team members were not involved enough or that they were overstepping bounds?

- **It's hard for me to say in terms of overstepping bound. But there were probably team members who weren't involved enough to no fault except resource constraints. I think that maybe overstepping maybe in terms of letting people 'be creative'. In some things I think there were constraints put on in areas there shouldn't have been. And perhaps hurt productivity a bit. This is kind of just a feeling. In some of the other teams, it's been a little looser in 'here's your stuff go do it' and in this team it was 'you need to do stuff this way'. (Kind of a role thing)**

Were you typically aware of what other team members were working on?

How did you communicate with other team members during the project (e.g. email, walk to cubicle, phone call, IM)

- Probably for me, more face to face. Face to face and email. Obviously for folks that are here I'm a face to face person. ZBut some team members weren't located here.

Did you ever have a disagreement with another team member(s)? How was this resolved?

- I'm not sure that they were resolved. I think that thins went on and things got done. I'm not sure that those things got aired out. Towards the end of the project I was less and less involved too.

Do you feel you received the support you needed from other team members to carry out your role in the project?

- Yeah.

How well overall do you feel the team collaborated? With respect to other projects you've been on?

Usability

How well did the usability engineer work within the team?

- I think she was terrific. I think this particular project if we didn't have a usability engineer, it would have taken a hell of a lot longer. We may not have met our goals that we needed to. I think in the beginning it took a little bit to get the coordination down.

Is there value in having a separate usability engineer role on the team? What did the usability engineer contribute to the team?

- It'd be nice to have a usability person on every project we're on. I appreciate the fact that she was open to suggestions to suggestions about the UI.

Did you understand the sorts of things that the usability engineer did and why?

-

What was your relationship with the usability engineer?

- I mean I communicated via meetings and email. And I tried to comment on things as they came out as I had the time to do so.

How well did you communicate with the usability engineer?

How important were usability evaluations in the conduct of the project?

- I think they were pretty important. I don't quite remember exactly what came out of them but I think we learned some stuff from them. It would've been nice to do more with real customers.

How does this compare to other projects you've been involved with?

- We barely, I don't recall if we've done anything like that before, now that I think about it. We've done beta which incorporates all kinds of testing but we've never monitored people in terms of usability.

What benefits/problems were encountered by incrementally developing the user interface? How does this compare with past projects?

- Pluses is we ended up with a better product, I think. I don't know whether it slows down the development, it may. But in the end it probably saves time. Doing it the other way may be quicker but you have to make fixes/changes in the next release so in the long run it saves time. I think we'll have happier customers because they're going through these changes with us and are a part of the changes we're making.

Did incremental develop affect the overall quality of the user interface? Compared to designing the UI up-front?

What was the role of the usability tests with regard to the overall product? What did they contribute if anything? How does this compare to past projects?

Did you believe that usability design decisions were made with respect to high level goals of the project? Were any changes not aligned with high level goals?

- Absolutely, yes.

Were there conflicts between team members with respect to which usability changes that need to be made? How were these resolved? (tradeoffs?)

- I think there were discussion. I think they were healthy. I would call them healthy discussions more than conflicts. I think in the beginning it was more tense until everyone learned how to work together. In this project, I don't think any of those people worked together, so that makes it much more difficult.

What was the effect of having usability and development working in parallel? (generally having the usability engineer work 1 iteration ahead of the developer) What were the benefits? What were the problems?

- In some cases some of the handoffs were difficult because things were changing and <developer> had problems.

Were there conflicts between the UI design and development constraints? How were these resolved?

Information sharing/Project management

How often did you refer to usability documentation on the portal? How was it used?

- Not very often. Not because that's the right thing, but because of time availability. I think I more looked at usability docs more than anything else—you know the mockups and stuff. I know I looked at them (usability test results) once. I mostly rely on info from meetings and I ask questions.

Goals?

Scenarios?

Mockups?

Claims/design tradeoffs?

Iteration plan?

How did you share information with the team?

Did you encounter any problems sharing information with the team or getting information related to the project?

- **No.**

How aware were you of development progress throughout the project? Were you ever surprised by changes made to the system that you didn't expect?

- Sometimes. I can't give you precise things. Maybe not so much surprised as. I mean there were certain things where there were expectations but they didn't get met. There was some, I don't know if you'd call it conflict, but nonagreement on when things were going to get done. Project plan said one thing but reality was not in the project plan and I don't think it ever was in the project plan.

How did you use Team System if at all during the project? Did you notice any problems with how it was being used?

- A little bit, to look at tickets here and there. (in meetings and offline)

How important were the different types of meetings held during the project (daily, feature team, review meetings with customer)?

- I can't comment on customer meeting except that I would expect that they were important. Some of the meetings were a waste of time. Things got repeated and preparation was not good. I tended to not go to all the meetings. In the beginning it felt like we were not moving ahead. It took us a long time to get going.

How well did the team stay on schedule during the project?

- It depends on what you compare it to. Not too bad. Probably on par if not a little bit better (with other projects). I mean we fell behind but that's normal.

How would you compare project velocity to other projects you have been involved in?

What things were done to address problems with project velocity?

How often did requirements change throughout the course of the project? How were they handled?

How does this compare to past projects you've been involved in?

- Normal, if you compare it to other projects. If I recall, there were a couple times where promises were made that maybe shouldn't have been without maybe thinking through things. Like I said I didn't go to those meetings but the outcome would be 'wow we promised this how will we deliver that?' We should have been 'we need to go back and evaluate that'. That goes back to 'lack of project management experience.'

Others

What are your thoughts on the development process used for this project?

- Not really. I mean I think it was, based on where the project started, I was pleasantly surprised at the outcome. It seemed like at the beginning, it was doomed, I mean it really did to me, that's a tough word. It took a long time to get moving. And there were lots of obstacles to get over too. I think all in all they did a good job.

How does it compare to other projects you've been on?

What things would you have done differently?

Any other problems/issues you encountered?

Quality Assurance

General questions

What was your role in the project? What did you do?

- I was the tester. I tested the product both for functionality as well as clarity of usage and consistency.
- Varied. But I'd average 25 hours a week on it.
- <project> is the other one (project I'm on).

Project goals

What were the key design goals for the system?

- They wanted it to be easy to use, especially because it would be used by people who weren't very computer savvy. They wanted clarity of the information put into the system. And they wanted it to allow for quick entry of information. It was supposed to increase clarity of information as well as speed up entry time.

Do you believe everyone had a clear understanding of the goals? Were everyone committed to those goals?

- I think so. Yes (they were committed).

Were there goals that conflicted with one another? How were they resolved?

- No, I don't think so. I don't think there was every a time where we had to sacrifice on eof them for the goodness of others. I think there were times where it lookee like we might have had to do it, but we found a way around it.

Were these goals used to guide design decisions in the project?

- **Yes, yes. And a, those goals were always kept in mind, especially in the user trials where we would take the program and let people who weren't involved with the project try it out and see how it works. We would make sure there weren't parts of the program that were unclear. Like sometimes we would be able to tell from those trials what parts of the**

design weren't clear. We were able to figure out what actions were taking too many clicks or maybe weren't as clear as others. And we then were able to find out where the weaknesses of the program were in those three areas and then we could then make improvements.

Do you think the team met the project goals? Why do you believe that it has or hasn't?

- **Yes. I think because, since we were consistently meeting together and constantly revising what we were doing, and seeing ways to improve it, it was a lot easier to make course corrections along the way. And since we were constantly making improvement, looking for ways to make improvements, I think the product ended up being very easy to use, very quick to use and allowed for a lot of clarity because of all the different drafts that we went through, we came, by the end, to a really polished product.**

Did the team always have a common understanding of what things needed to be worked on and why?

- I think generally yes. I think on some issues there were disagreements about..in some cases the design and development I think had differences of opinion. Everyone was going for the same goals but obviously when you're dealing with a group of people, many people in trying to fix the same problem offered different solutions. And that caused delays in figuring out what we were going to do because of the different ideas involved.
- How were they resolved? --Generally when a problem would arise, we as a team would discuss it, offer our opinions, solutions. And then generally, there would be a majority on one side or another. Oftentimes, those who were outvoted would concede to the majority idea. Or if it couldn't be settled in that way, or if there were still differing opinions that were strong, we would discuss that issue with the client to resolve it. Because they were helping us with the design.

Customer relationship

How important was it to be in contact with the customer/client during this project?

- It was very important to be in contact with them at various point sin the process. Because there were lots of things where we had notes, or copies of forms—we had input from them---there were a lot of assumptions we made about what they wanted that needed to be clarified a little bit. Many times, we had a completely different idea of what they wanted as opposed to what they did want. Some cases it was not necessary to be in contact because sometimes they didn't have a lot of helpful input. In those cases it was more time-efficient to make a decision and present our decision to them for their approval as opposed to coming to them with our several different paths and having them choose.

What benefits were derived from interactions with the customer/client?

- Benefits were, we had more input more often and made course corrections as we went.
- The negative aspects were, that we became so dependent on customer opinion that we hesitated in making our own decisions and since we were the ones making the product we had more know how about how to make it best—but we would often hesitate to make those decisions because we wanted their opinion or their ok—that slowed us down.

How does the amount/quality of contact compare to other projects you've been on?

- In this project, there was a lot more. In the <other> product that I've been on, we have been basically on our own. I mean we're making all these decisions, we're fixing things on our own. And then in the end we'll present a finished product. I mean we talked to them at the beginning and at the end but in the middle there's not very much.
- In the touchscreen product, we were constantly in touch and that affected a lot of our decisions and a lot of how we finalized the product.

Did you feel that there should have been more/less customer contact? What should have been different with respect to this?

- To be honest, I think that the amount of customer contact that we had was good. I wouldn't necessarily change how much contact, but perhaps changed a little bit of how we viewed it. If we had been—a little bit more independently minded in making certain decisions and not be so dependent on the customer. But it was really good for them to give the ok on what we did decide. And that amount of contact was very helpful.

Team Collaboration

How well did team members collaborate with one another? Did you notice certain problems related to how well one person worked with another? What caused these problems?

- I thought in general everyone did a really good job of collaborating, and even though there were---we had one representation from each group (testing or design or development), there were lots of times outside the meetings where I would meet with the developer and he would show me the work he had done and I would show things I thought were problematic and he could fix them immediately or take my opinion. The usability representative was able to work with the developer to work on the layouts so they could be user friendly and they were done outside the meetings. Because we were working so close to one another, we were able to help each other through the design process.

What sorts of problems were encountered and what caused them?

- There were some but they weren't major. Oftentimes, there were differences of opinions. Usability thought the screen should be set up in a certain way but development thought they were more practical (more easier to develop) to be setup in another way. When both people had their reasons for wanting it one way vs the other, we just had to bring those discussions to the team and we would decide as a team what the best way overall would be to fixing the disagreement.

Did you ever feel that certain team members were not involved enough or that they were overstepping bounds?

- Generally, I think everyone had their say and , I think due to the size of the team. It was easy

for most people to have their say, to be able to contribute in that way and then. I say everyone was pretty much able to fill their roles.

Were you typically aware of what other team members were working on?

- Yes. Because often those things would be brought up in the meetings or after the meetings or before the meetings. The collaboration that would go on, in terms of bouncing ideas of each other, before we start committing a lot of work to a certain path... it was definitely something where we knew what each of us was doing.

How did you communicate with other team members during the project (e.g. email, walk to cubicle, phone call, IM)

- I'd say face to face as well as email.

Did you ever have a disagreement with another team member(s)? How was this resolved?

- Given that my contribution to the project was being a software tester---there wasn't much that I created so I didn't have a ---the opinions that I gave were ways of fixing others' contributions. I was just giving my opinion on someone else's work. I wasn't development or usability so there wasn't too much disagreement that came from that. But I would try to improve the product the best I could and sometimes that meant offering suggestions/improvements on what other's had done. But there weren't huge disagreements.

Do you feel you received the support you needed from other team members to carry out your role in the project?

- Yeah, yeah. Whenever—one of the parts of our project was having a test case review—everyone would come to those and give opinions and point out things that I needed to work on. So people helped me to do my job.

How well overall do you feel the team collaborated? With respect to other projects you've been on?

- I would say we collaborated pretty well. I mean, lots of people had strong opinions. Overall, we did a good job of working through those differences.

Usability

How well did the usability engineer work within the team?

- I think she did a great job. The work that she did I thought added a great deal. I mean it was really, really easy to use. It was streamlined, it ...you could do it really really quickly. And I think that added a lot to the project. To me, it added excitement to the project, it was something that we could...it felt like we were doing more in the first half than in the second. And I think it turned out to be a really good looking program.

Is there value in having a separate usability engineer role on the team? What did the usability engineer contribute to the team?

- I think so. And I think that was best shown because, when the original concept for the

touchscreen was brought up. I envisioned a lot of picklists on one page. The way I saw it would've been very cluttered—it wouldn't have been attractive. It would've been functional but hard to use—caused a lot of errors. Having someone experienced in how to make something user friendly—the final product just worked a lot better and fulfilled the customer's expectations a lot more than my idea would've been, or other team member's would've been. Some people—just because of their personalities— are used to putting things in a non-usable way—but the majority of people need something that's easy. It helped cater the system to customer's needs who maybe aren't very computer savvy.

Did you understand the sorts of things that the usability engineer did and why?

- I don't know very, very much about the questioning that happened at <customer company>, but certainly in the usability studies I definitely saw the merit in that. I helped participate in that and there were a lot of things that when she was designing it, we made a few assumptions that something would be clear and we found out that it wasn't. The mockup helped in that it was something that could guide the developer—they were able to follow something that we had already looked at, and it was something we could actually apply what we wanted to do and apply functionality to it.

What was your relationship with the usability engineer?

- I helped out during the usability studies, because the program was setup in a way, we didn't have a touchscreen for the users to use. We were simulating a touchscreen by helping simulate a regular touchscreen. And my job was to simulate the touches. Also, when I was looking at the mockups I would have interactions with the UE because there were certain things that in testing—I wanted to know about the functionality.
- Since I was writing my test cases and preparing in how I was going to test the program, I wanted to know what the UE's vision was for how the user would interact. And sometimes when I looked at the mockup, I would offer suggestions for things that might be improved on.

How well did you communicate with the usability engineer?

- Yeah, yeah. Generally through email because I would see things and then make notes of things I just thought could be a little clearer. Or sometimes we would have meetings. A couple of us would get together and point out things that might not be as clear. We wouldn't do this many times, but just do this for things that needed to be clarified.

How important were usability evaluations in the conduct of the project?

- I thought it was very useful because we were able to experiment our program on various types of people. One of the people that tried it out was someone who had lots of experience with touchscreen interfaces so he was able to give us lots of feedback about how we could make our product better. Other people weren't as familiar with those types of applications, we got a lot of feedback from them b/c we could see the things that they would struggle with. Lots of those usability things that we thought were obvious but that people who weren't familiar with struggled with.

How does this compare to other projects you've been involved with?

- I'm relatively new to the company, so as far as usability testing—we've been doing testing here but mostly not usability testing. So no, this is the most amount of usability testing. The first time that I'd experienced that. And I thought it was very helpful to get feedback on it.

What benefits/problems were encountered by incrementally developing the user interface? How does this compare with past projects?

- I think that making it in pieces was helpful but I think one of the problems that came from the idea of the idea of going through it incrementally, at least me as a tester, was that when I would try to write tests to test the way that the program would work, since we were doing it incrementally, we would make changes all throughout the project which ultimately made the product easier to use and better but made it difficult for Qa to make tests that would be specific to the product as well as potentially cause problem for documentation that are trying to write out the workflows because sometimes the workflows would change. That made it difficult for me, and made it difficult for development but I think constantly changing and molding it to what the customer needed made it a better product.

Did incremental develop affect the overall quality of the user interface? Compared to designing the UI up-front?

- See above

What was the role of the usability tests with regard to the overall product? What did they contribute if anything? How does this compare to past projects?

See above

Did you believe that usability design decisions were made with respect to high level goals of the project? Were any changes not aligned with high level goals?

- Yes, definitely. We were always keeping those high level goals in mind. So after the usability testing that we did, we saw if we weren't fulfilling those goals we needed to make changes to the product to make it work better.

Were there conflicts between team members with respect to which usability changes that need to be made? How were these resolved? (tradeoffs?)

- See above

What was the effect of having usability and development working in parallel? (generally having the usability engineer work 1 iteration ahead of the developer) What were the benefits? What were the problems?

- I thought it was good b/c the change would be proposed, development would fix it, and then by the next meeting we could see the implementation of the new change the downside is that a lot of work was done, and then redone b/c we were constantly making changes. Maybe it could've caused more unnecessary work-but it was also good to see the implementation of these things and test them a little bit on the users.

Were there conflicts between the UI design and development constraints? How were these resolved?

- Yes, there were some. A lot of times where as we were constantly trying to polish the program, and make it more user friendly, faster easier all those things those high level goals. There were a few times when development had the complaint that this was something that, given the amount of work required to fix it, there were constraints on whether or not that would be implemented. Just b/c, development would weigh how much we would gain from the change, as opposed to how much time would be required in development to implement it. If it was something that usability felt very strongly about, we would discuss it as a team and decide whether or not it was something we needed to implement and sometimes as a group we would brainstorm a way we could make a similar improvement made by usability that wouldn't require as much time to implement.

Information sharing/Project management

How often did you refer to usability documentation on the portal? How was it used?

- I personally used the mockups a lot. As soon as they were updated, I would print them out and write out the functionality so I would know how to write my test cases. The <customer company> documentation was not as necessary b/c my testing was based on what was produced by usability and development. My work didn't depend directly on <customer company>.

Goals?

Scenarios?

Mockups?

Claims/design tradeoffs?

Iteration plan?

How did you share information with the team?

- Email. And oftentimes with development I would send an email and then we would meet face to face. Whereas with usability, the Ue was there 3 times a week so dev was there the entire week. If I needed something I would walk to his cubicle.

Did you encounter any problems sharing information with the team or getting information related to the project?

- **Generally, not.**

How aware were you of development progress throughout the project? Were you ever surprised by changes made to the system that you didn't expect?

- Generally, not just because development was oftentimes---as we'd bounce ideas off each other---when he would implement something he would invite me to see it. Generally, we all

kept in close contact.

How did you use Team System if at all during the project? Did you notice any problems with how it was being used?

- I used TS as a tester to write bug tickets-things that I saw-I would then assign them to the developer to fix or to usability if there was some graphic error or something I thought should be clarified. Then they could implement them and then I could test them to see if they'd been fixed.

How important were the different types of meetings held during the project (daily, feature team, review meetings with customer)?

- I thought depending on what the subject matter being discussed was—some meetings were more effective than others. Many meetings involved looking at what we had done, updating everyone on where we were—I thought those meetings were generally effective. Other times, some meetings we would meet together and not discuss what we were working on, but we would constantly look at the calendar and see where we were at the calendar and those meetings weren't as effective cause we didn't discuss things that would help us make the product. The feature team meetings I thought were good b/c we were able to go through the product and see what dev had come up with as a group.

How well did the team stay on schedule during the project?

- I think generally, we did pretty well. In some cases we got behind just because of unforeseen changes that development had to deal with—often times dealing with third party software.

How would you compare project velocity to other projects you have been involved in?

- It was faster than average, I'd say because—due to the small number of people—how close we were. It was very easy to spot an error, rely that to dev, dev could then change it in a few hours, it could be tested and fixed and there were a lot of those errors would've taken a lot more.

What things were done to address problems with project velocity?

How often did requirements change throughout the course of the project? How were they handled?

How does this compare to past projects you've been involved in?

Others

What are your thoughts on the development process used for this project?

I think it went well.

How does it compare to other projects you've been on?

What things would you have done differently?

Any other problems/issues you encountered?

Quality Assurance (Acted as consultant to team)

General questions

What was your role in the project? What did you do?

- QA consultant (mentor to <qa person>)
- I monitored Qa work items that <qa person> had. Try to make sure we had the right amount of coverage for what we were testing. Participated openly in the meetings regarding open issues, fixes and timeline sort of things.
- 1-2 hrs per week on the project.
- <products>

Project goals

What were the key design goals for the system?

- Easy to use, straightforward client interface for work order and notification data entry for <BMS>.

Do you believe everyone had a clear understanding of the goals? Were everyone committed to those goals?

- I would say yes. For the most part.

Were there goals that conflicted with one another? How were they resolved?

- I'm not aware of any, other then our resource and time constraints. Those basically conflicted with the main goals of the project.

Were these goals used to guide design decisions in the project?

- Sort of. We also used a lot of customer input as well. It was kind of like a 50/50. Maybe if we didn't get a particular design constraint from a customer we fell back on our high level.

Do you think the team met the project goals? Why do you believe that it has or hasn't?

- **Yeah, for the most part. Maybe not the time goal. Definitely the project goals. Yeah, I think we overshot our time on that a little bit (both phases). Not badly, but...**

Did the team always have a common understanding of what things needed to be worked on and why?

- From my perspective, no. I would say that <developer> knew more about what need to be done and what needed to be tested as well.

Customer relationship

How important was it to be in contact with the customer/client during this project?

- I mean, I liked the level of involvement we included the customer in. I just think that maybe we, ... I'm not sure the customer knew what he wanted and we didn't know what the customer wanted either. And it sort of helped out in us getting the type of product the customer wanted. It seemed like sometimes we would go back and forth on issues and sometimes design constraints would restrict the ability to do certain things.

What benefits were derived from interactions with the customer/client?

- They were able to follow our process, our development process, so I think that they were there every step of the way. There weren't any surprises. We were also able to resolve issues before implementation of the product.

How does the amount/quality of contact compare to other projects you've been on?

- It doesn't compare. I mean I don't know. I've never had another project where we included the customer throughout the design process.

Did you feel that there should have been more/less customer contact? What should have been different with respect to this?

- I think the level of customer contact was good. I think it could have been organized better to be more efficient.

Team Collaboration

How well did team members collaborate with one another? Did you notice certain problems related to how well one person worked with another? What caused these problems?

- I'd say very well for the most part. I Just think there was a lot of redundant time spent

between usability and development. It seems like there was a lot of unnecessary churn. It seemed like we were changing low level pieces of functionality based off of user studies and the user studies in my opinion—they didn't eliminate bias—but they didn't include users that were familiar with the product. And when they were familiar with it they would use it in the least amount of time possible. And we made things different, that we could've left.

- It seemed like for a while there, for phase 1, we weren't getting anywhere with our meetings—and I don't know if there was a learning curve in the software.
- And things were changed in the end, visual things,...whether or not that was really necessary. Also, the review page, I distinctly remember a lot of discussion around how that should work, whether the user should be able to click certain areas so they can go back and change certain areas.
- The way that the interaction was handled was poorly managed as well, in my opinion. It seemed like we needed <qa manager> to get in to handle some of these design squabbles.

What sorts of problems were encountered and what caused them?

Did you ever feel that certain team members were not involved enough or that they were overstepping bounds?

- **Myself, but I didn't have a choice in that. Falls back on the time constraint and availability of resources.**

Were you typically aware of what other team members were working on?

- Not in the beginning,... I had to make a request for a feature meeting where we'd go over work items to make sure we know where everyone's at—through the help of <qa manager>'s management style.

How did you communicate with other team members during the project (e.g. email, walk to cubicle, phone call, IM)

- I would say, face to face or email. I mostly interacted with the developer and <qa person>.

Did you ever have a disagreement with another team member(s)? How was this resolved?

- The only thing that sticks out in my mind is the way that usability test is conducted. Once again, the data that was collected. Some of it was good data, but... I don't know that the correct people were tested and the test was executed correctly.

Do you feel you received the support you needed from other team members to carry out your role in the project?

- Absolutely.

How well overall do you feel the team collaborated? With respect to other projects you've been on?

- See above

Usability

How well did the usability engineer work within the team?

- Well. (wasn't involved in another team with a UE)

Is there value in having a separate usability engineer role on the team? What did the usability engineer contribute to the team?

- Yes. Because a lot of times people don't consider usability and user interaction with an interface when they design and develop products. There can be learning curve snags that you can reduce.

Did you understand the sorts of things that the usability engineer did and why?

- Yes.

What was your relationship with the usability engineer?

- Only in team meetings. Email as well.

How well did you communicate with the usability engineer?

- No.

How important were usability evaluations in the conduct of the project?

- See comments above. I would say, medium. I mean, I almost want to say. I mean the customer interaction could have provided us with a usable interface. But the UE provided us with a more usable product, that will require less training.

How does this compare to other projects you've been involved with?

- (see above) – first project with UE

What benefits/problems were encountered by incrementally developing the user interface? How does this compare with past projects?

- I think there may have been additional value if there was a complete comprehensive design. Cuz I feel like things were change din the later modules that also had to be changed in the earlier modules and if they were considered in the beginning those changes wouldn't need to be made.
- Now, we just discuss it as a team. My involvement is sparse, to say the least.

Did incremental develop affect the overall quality of the user interface? Compared to designing the UI up-front?

- See above

What was the role of the usability tests with regard to the overall product? What did they contribute if anything? How does this compare to past projects?

- See above

Did you believe that usability design decisions were made with respect to high level goals of the project? Were any changes not aligned with high level goals?

- For the most part. If time or schedule was a high level goal, then the extra churn would be something that would affect that. But I mean you have to weight that out.

Were there conflicts between team members with respect to which usability changes that need to be made? How were these resolved? (tradeoffs?)

- All the time. Specifically between the UE and the project manager. Resolved through team discussion or design/time constraints (project level and developer/qa time and usability constraints).

What was the effect of having usability and development working in parallel? (generally having the usability engineer work 1 iteration ahead of the developer) What were the benefits? What were the problems?

- I would say problems would be in implementing new functionality new iteration, you introduce new bugs as well. You create more work for the developer.
- The plus would be getting to a well molded, streamlined user interface that provides a great value for ease of use.

Were there conflicts between the UI design and development constraints? How were these resolved?

- Yes, absolutely. I would say research and product knowledge and compromise. If it's an <bms> constraint, we'd ask <researcher> about certain things.

Information sharing/Project management

How often did you refer to usability documentation on the portal? How was it used?

- Almost, never. If there's a document displayed on the screen during the meeting, I read it. If there's mockups that's displayed, I would get those. I felt like the mockups were more for development. Once we do the demo, and go over everything, in conjunction with the mockups, we know what we need to test for. They're usually there as a point of reference. Usually everything is discussed within the team meeting. If there's something that's not brought up in a meeting, I refer to a design doc/mockup and then the product manager.

Goals?

Scenarios?

Mockups?

Claims/design tradeoffs?

Iteration plan?

How did you share information with the team?

- During the meetings, email

Did you encounter any problems sharing information with the team or getting information related to the project?

- **No, see above**

How aware were you of development progress throughout the project? Were you ever surprised by changes made to the system that you didn't expect?

- I mean, yes but small level stuff. Nothing big.

How did you use Team System if at all during the project? Did you notice any problems with how it was being used?

- I used it to review work items with <qa person> and to make sure work items were closed properly. I know that there were reference docs available through team system as well.

How important were the different types of meetings held during the project (daily, feature team, review meetings with customer)?

- I would say that most of the daily meetings were not very efficient. The creation of the feature team meeting I guess was to try to get away from that unnecessary churn that was occurring in the daily meetings. It seems to be working well now. The meetings with <customer representative> provided great benefit. We essentially demoed the product form they and got feedback so I'd say they were very efficient as well. With the level of changes that were made, it seemed like sometimes there was a bit of churn from the customer interaction as well. In which case, it may be better to have the customer involve later in the dev cycle, or maybe part of the way through, and then maybe at the end. It's difficult to say what level of customer interaction there should be.

How well did the team stay on schedule during the project?

- See above.

How would you compare project velocity to other projects you have been involved in?

- Terminal. For a while, we were doing a demo every week which is kind of crazy.
- It's different from any project that's internal. Product management handles customer interaction and demos (typically at Meridium). Demos at Meridium are used for

development/design/QA so we can reduce bugs and issues that we have with it.

What things were done to address problems with project velocity?

How often did requirements change throughout the course of the project? How were they handled?

How does this compare to past projects you've been involved in?

- Churn refers to wasted time with minimal results.
- Other projects are minimal churn. Things get done. They're better managed. There's a set agenda for each meeting where high level things such as work items that are open are gone over and discussed and actions are taken at that point. Discussions can also bring about an action item. With product management and development and QA, some sort of action is then taken to move forward.

Others

What are your thoughts on the development process used for this project?

- Probably the most difficult thing to do was to manage the level of usability and customer requirement interaction with the development process. Increasing that efficiency would provide a faster and more suitable product for the end user. I don't see, from my standpoint, that the agile development process is the correct way to go. I think if you have a more well defined, set process, in the long term it will be more efficient.

How does it compare to other projects you've been on?

What things would you have done differently?

Any other problems/issues you encountered?

Documenter

General questions

What was your role in the project? What did you do?

- I was the documentation consultant. We were going to decide if documentation was needed. Ultimately we decided it was not.
- I worked with <usability engineer> and <qa person> –ensuring tha the interface worked in such a way that the user would not have any questions.
- 2-3 hours a week on this project.

- Also was doing documentation for <product>.

Project goals

What were the key design goals for the system?

- To create the touchscreen kiosk system. To develop the system so people at <customer company> could use it.
- Wanted it to be very streamlined and easy and quick to use. It should have an intuitive user interface. Talked to <customer representative> a lot to make sure it worked properly for them to use.

Do you believe everyone had a clear understanding of the goals? Were everyone committed to those goals?

- I think so. We talked about that the first few meetings. We spent a lot of time on the interface, talking about what we wanted the goals to be. Spent time going over <usability engineer>'s powerpoint, talking about the goals.
- I think everyone was committed. In discussions about them, we chose goals that everyone felt was important, so naturally would be committed to them.

Were there goals that conflicted with one another? How were they resolved?

- All facets of the same thing.
- The only conflict, was we spent a lot of time trying to make it work for <customer company>. Working to make it more open so other companies can adopt it. (If we decide to make it a product)

Were these goals used to guide design decisions in the project?

- **I definitely feel like they were. Those were the questions we asked ourselves when we decided to make a change.**

Do you think the team met the project goals? Why do you believe that it has or hasn't?

- I really think they did from a design standpoint. The software is very intuitive, everything jumps out at you. I think everyone can pick up and use it, which I think was the point.

Did the team always have a common understanding of what things needed to be worked on and why?

- Everyone had a common understanding of design goals, but sometimes there was conflict as to what was most important to do next or what should be done to accomplish a design goal.
- From my perspective, sometimes there was documentation conflict. <project manager> wanted documentation. Documentation would make things in the UI less intuitive. E.g. adding documentation links to the interface would make it cluttered. Initially, I felt the software should have something, but lots of little popups would have made it more confusing. Would have preferred a unified help system. Eventually we decided not to have

documentation at all.

Customer relationship

How important was it to be in contact with the customer/client during this project?

- In the context of a custom solution, I think it was incredibly helpful. Got input about mockups directly. We were in an immediate position to change things. It was great for <customer company>.
- In a general sense, it was not as good. I think it was still helpful because we were talking to someone who would use it in a real world. Usually, working with a consultant we would not get feedback from people who actually use it.
- In the other <PRODUCT> project, we were talking with consultants who were working directly with customers but weren't users themselves. From a design standpoint, we only had consultants saying things based on feedback on the previous software. We couldn't change midstream because we weren't talking directly with customers.

What benefits were derived from interactions with the customer/client?

How does the amount/quality of contact compare to other projects you've been on?

Did you feel that there should have been more/less customer contact? What should have been different with respect to this?

- Never dealt with a customer in a 'more contact' scenario. Talking with them more than every week would have been 'cluttered'. In generally, more than that we would be making changes too rapidly.
- Less contact we would be in a similar situation, we would have not been able to make changes as quickly.

Team Collaboration

How well did team members collaborate with one another? Did you notice certain problems related to how well one person worked with another? What caused these problems?

- More perhaps than in <PRODUCT>, they were very open in their opinions in how things should work. People would argue about how things would work. But that openness was helpful. Decisions were made based on arguments that ultimately made it better.
- <project manager> and <qa manager> would periodically argue with each other but as a group we made a decision based on what <customer company> would like. Sometimes devolved into what <project manager> would like or <qa manager> would like. Ultimately, they were resolved as a group, this was the best way for <customer company>. <project manager> was

focused more on <customer company> early on. <qa manager> was focused more on 'how can we make this so we can make it a product later so that other people can use it'.

What sorts of problems were encountered and what caused them?

Did you ever feel that certain team members were not involved enough or that they were overstepping bounds?

- **Sometimes I felt like I was not involved enough. That is because myself and <project manager> and <documentation mgr> decided there was no need for documentation and I didn't need to go to meetings as often.**
- **Most information was from talking to <project manager> or <qa person> or <qa manager>. I was not doing enough in the product to drive it. I was involved early on making sure the interface was developed properly.**
- **Usually there was a clear line between departments and what they should be doing.**

Were you typically aware of what other team members were working on?

- I felt like it was talking to members of the team about what they were doing and what was going on. To see if anything needed my input.

How did you communicate with other team members during the project (e.g. email, walk to cubicle, phone call, IM)

- Mostly emails and walking up to them. Mostly I would email <project manager> or <developer>. <qa person> spends his time at his desk and I would ask him as I passed by.

Did you ever have a disagreement with another team member(s)? How was this resolved?

- Disagreement with <project manager> about how help should be integrated into the software. Problem with <web technology> is you can't have help like we do with rest of the product. I would have to write the help and <developer> would have to put it in himself. <usability engineer> proposed that it would be possible to put help bubbles in different places. Any place where it would not be immediately apparent what the user had to do. I felt that solution was too cluttered. Eventually, several discussions with <documentation mgr> and <project manager> and occasionally <usability engineer> with what would look and work best for a user. Ultimately decided if point is to make it intuitive there would be no need for help.

Do you feel you received the support you needed from other team members to carry out your role in the project?

- Yes. Anytime there was a question or point of contention. There was a great deal of openness.

How well overall do you feel the team collaborated? With respect to other projects you've been on?

- Better later on. At the beginning there was more arguments and disagreements. Mostly arguments with how the system would work best for the user. Even with arguments everyone

was on the same page with what we were trying to do.

Usability

How well did the usability engineer work within the team?

- I think she did very well. Every meeting, she had new mockups, the design at the beginning. She did a lot of work creating the interactive powerpoint so even before <developer> started coded people could click through it and see how it would work. Early feedback.
- She did quite a lot of work trying to make it as easy as possible. Coming here on Tuesdays and thrus when she wasn't supposed to.

Is there value in having a separate usability engineer role on the team? What did the usability engineer contribute to the team?

- I think it was a huge help. QA wants to work from a user standpoint, but they're more focused on it working correctly, not on that it's usable.
- Doc wants to make sure it makes sense to a user—the user knows what to do at a given point, not whether it's useful/usable. How useful is it to a person. She had different styles of mockups and pros and cons for each one. Driving the very early stages.

Did you understand the sorts of things that the usability engineer did and why?

- I think so. Run throughs with How x employee would work through the software. We look at that at the beginning and it was helpful. Looking at interviews from <customer company> site visit and basing work on that was very helpful.

What was your relationship with the usability engineer?

- We would email some if I had questions about what she meant in a document or what the workflow was supposed to be. Not necessarily that close.

How well did you communicate with the usability engineer?

- If I had a question I would email a question and relatively quickly she would respond. She didn't have as many questions for me.

How important were usability evaluations in the conduct of the project?

How does this compare to other projects you've been involved with?

- I think those were one of the most important parts. Going to people who didn't use it before and seeing how they would use it. Helpful in determining how we did things.
- There was usability testing but not in the same way. We would test that it would work in the way that consultants told us it should work. But there was not the same kinds of tests until we had the demo with <customer> in the end. We couldn't see how employees would use it. This was one of the biggest problems. 3.2.3 worked fine from our perspective but for people using it all day there were problems. Didn't have input from customer during 3.2.3 development. Didn't have feedback until <newer>.

What benefits/problems were encountered by incrementally developing the user interface? How does this compare with past projects?

- Changes could be localized to within an iteration. It was definitely helpful in that way. The biggest problem is not the agile process itself, the problem was with semi-hard release dates. You can't do agile with that sort of deadline. Especially when you can talk to user on a regular basis, agile was incredibly helpful.
- <product> was 'agile' but not really. They had iterations and the idea was to do UI/development incrementally but especially near the end, there was more general fix things all over the project. It did not work as well because we could not do things as quickly because it was not localized changes, they were all over <product>.

Did incremental develop affect the overall quality of the user interface? Compared to designing the UI up-front?

- It may have. I'm not sure if I could know that. It helped in that we could show off one part of something and get immediate feedback. So it helped in that over time the entire product had discussion with the customer, and making it all function to a real world standard. Each individual part was better which made the product as a whole better.

What was the role of the usability tests with regard to the overall product? What did they contribute if anything? How does this compare to past projects?

- See previous questions.

Did you believe that usability design decisions were made with respect to high level goals of the project? Were any changes not aligned with high level goals?

- I definitely think so. Every time someone proposed an idea or wanted to change something we asked ourselves does this suit these goals.
- Sometimes we asked how difficult a transition will it be to the forms to this computerized system. <usability engineer> was one of the biggest people asking that question. Will they understand?

Were there conflicts between team members with respect to which usability changes that need to be made? How were these resolved? (tradeoffs?)

-

What was the effect of having usability and development working in parallel? (generally having the usability engineer work 1 iteration ahead of the developer) What were the benefits? What were the problems?

- I think that was incredibly helpful, especially for an agile process. Then you can work in tandem, usability working on the very next thing. Speeds up the process of designing it. For tailoring it specifically for <customer company>. The employee list—can we populate based on the plant field.

Were there conflicts between the UI design and development constraints? How were these resolved?

- I don't think that any problems came up that I'm aware of.
- I think there was the potential problem of usability designing something that wouldn't work for development. I don't think it happened cause there was good communication. More than working on mockups, <developer> talked to <usability engineer> about how things worked.

Information sharing/Project management

How often did you refer to usability documentation on the portal? How was it used?

- I think it was very useful. I looked at it whenever I looked at the mockups. <usability engineer> had the instructions of the workflow from the very beginning. Made sure you could use it in the way that followed the mockups.
- We looked at documents at meetings and they were there for us to look back on. Feedback from users (comments they made) during tests.

Goals?

Scenarios?

Mockups?

Claims/design tradeoffs?

Iteration plan?

How did you share information with the team?

- The only information I really shared was my perspective of the usability to <usability engineer> and <qa person>. My opinions on how things should work from a usability standpoint. Design in such a way so that there would be minimal documentation.

Did you encounter any problems sharing information with the team or getting information related to the project?

How aware were you of development progress throughout the project? Were you ever surprised by changes made to the system that you didn't expect?

- Nothing I can think of offhand. Any time major changes were made, we had already discussed it.

How did you use Team System if at all during the project? Did you notice any problems with how it was being used?

- I didn't use it more than once or twice for the project because I didn't write any tickets. Mostly talking about things others had said, which may have been captured in a ticket, I don't know.

How important were the different types of meetings held during the project (daily, feature team, review meetings with customer)?

- I'm not sure how useful the daily meetings were. I feel like the prep meeting with <customer representative> and the feature team meeting where we talk about what <customer representative> said were important. Figure out what we'd done, what we do next.

How well did the team stay on schedule during the project?

- Compared to <PRODUCT>, extraordinary. I feel like there were times where we didn't meet the timeframe we wanted (which is the risk in doing agile). The whole point (of agile) is that you do it right the first time. There was lots of problems (in <PRODUCT>) that with discussions with real customers, they would know they were designing in a way that was not very useful.
- I do not feel that it is too important to follow a stringent schedule in this way. Cause it would take a lot longer for the next release.

How would you compare project velocity to other projects you have been involved in?

What things were done to address problems with project velocity?

How often did requirements change throughout the course of the project? How were they handled?

How does this compare to past projects you've been involved in?

- I just feel like in the beginning we were planning to do documentation and wanted to do the start page a certain way but after doing testing we did it another way (not have documentation). I think that helped the project, making the shift. There was no dramatic shift in the UI. There were several changes in the way it worked that added up to large changes.

Others

What are your thoughts on the development process used for this project?

- I think it's much much project then others. Especially waterfall where you design everything at the beginning and then use those artifacts during development. That's better if you're hitting a specific deadline. Which is why a lot of feature teams here at meridium use that system. But it's not as useful ultimately because you can't change midstream.

How does it compare to other projects you've been on?

- I feel like the openness of the members helped in the end. That kind of openness starting with no predefined way to do things helped. With meridium core/software you work in a particular workflow so there's not much leeway in how the workflow is designed. Helped us design a very usable product.

What things would you have done differently?

Any other problems/issues you encountered?

Project Lead

General questions

What was your role in the project? What did you do?

- I was the development lead. The team lead.
- Planning, scheduling, monitoring the work items. Staying on top of technical issues. Coordinating reqs. Gathering with the user.
- Maybe 15-20 hrs a week.
- Various university activities, the work item <work ticket system> to team system transfer.

Project goals

What were the key design goals for the system?

- User satisfaction, speed which was a big one, and collecting better data, good data.

Do you believe everyone had a clear understanding of the goals? Were everyone committed to those goals?

- Yeah, to some degree.
- To some degree. When you're in the middle of something, you have other things going on and you don't prioritize those goals sometimes when you want to get functionality then those goals are not at the top of your priorities. But they were all aware of it and really did strive towards it.

Were there goals that conflicted with one another? How were they resolved?

Yeah, you have to balance. Providing a slick nice interface so that it's highly acceptable to the user might conflict with the speed (user was supposed to enter data in a minute or two). And making really nice looking interface might not provide a fast interface. (To resolve them) you try to reach a happy medium.

Were these goals used to guide design decisions in the project?

Somewhat. Sometimes not (e.g. Management decisions that we wanted to use <web technology> because that's where the company might want to go and we wanted to use this as a test project.)

Do you think the team met the project goals? Why do you believe that it has or hasn't?

- Yes, but it's hard to say without having had take to the end users to try it b/c we haven't gotten ok by <customer company>. They're stalled.
- Because it is from what testing we can do, it seems in the ballpark of the speed that we want. It is a nice interface. It's simple. It's pleasant to look at. It's intuitive.
- The basic design of getting the information from <BMS> is limiting incorrect data from being input. The design of the interface doesn't allow you to enter incorrect data. It was designed with that in mind.

Did the team always have a common understanding of what things needed to be worked on and why?

- **On a scale of 1 to 10, yeah they did but there's always issues.**

Customer relationship

How important was it to be in contact with the customer/client during this project?

- Crucial.

What benefits were derived from interactions with the customer/client?

- Quick feedback so we didn't look down the wrong path. We could ask him if he liked this or this so we could check things out. And ask him questions when we didn't understand. And it assured me/us that we were doing the right thing all along. As opposed to building it in a vacuum. It increased confidence.

What problems?

- Constantly changing requirements. You can't do planning very well.

How does the amount/quality of contact compare to other projects you've been on?

- Extremely high. Extremely, extremely high.

Did you feel that there should have been more/less customer contact? What should have been different with respect to this?

- Yeah, by now we should've had end users test it but that's not. Because of the nature of the project, we need to actually work on their systems and that's what has it stalled. So we have to get on their <BMS> system and that's what made the whole process stalled. But that seems kind of natural. Par for the course like dealing with an <BMS> system. It's a huge security issue too.
-

Team Collaboration

How well did team members collaborate with one another? Did you notice certain problems related to how well one person worked with another? What caused these problems?

- On the days that people were here, I would say hourly. I communicated with team members frequently a few times a day. Different people. And other people were communicating within the team. I think it would've been greatly improved if we had actually been collocated—in a single space. (in the agile sense).
- They recognize that this is an important issue, so they moved everyone around and put the <product> team together.
- A major issue has been the significance of the project. It was not a significant project.
- At the beginning <usability engineer> and <developer> didn't get along. Their resolution was odd. Things like <usability engineer> defined things in very precise detail instead of trying to work with him to make decisions which is how I would've thought it would've been done, but everything was an argument so she just documented how things should be done and he actually started to follow it so it was resolved, I guess.
- Long time employees are used to doing things one way and that made things difficult. Especially when we didn't have a clear idea of how we should clear things. Even though we were supposed to do something like the Simon agile process which isn't much of an agile process. An ill-defined process was not helpful and that isn't to say we didn't know what the process was. It's to say, people don't think it's important to understand the process. <developer> said it wasn't important to talk about the process.

What sorts of problems were encountered and what caused them?

- The utter lack of experience on the team caused major problems. The very strong personalities that insisted that things should be done their way cause they knew what they were doing caused problems.
- How did you deal with them?
 - o Tried to decide what should be done and explain that to people. Explaining things too. Either we could've gone into every meeting having fights or try to direct people instead of telling things what to do. And there was only one developer. If there had been a counterbalance to be different things would've been different. If <developer 2> had actually been there, there could've been some counterbalance to <developer> and his lack of experience. The project turned out well. That's the bottom well.

Did you ever feel that certain team members were not involved enough or that they were overstepping bounds?

- Yeah. <product manager> just never—half the time he was never there and half the time he was there he was doing something else. It was very irritating. And when I talked to <management> about it he said he was on a bunch of projs. And ours was not a priority. <developer> constantly went over my head to talk to <management> and other people to make design decisions. He would make decisions on his own and whne I tried to direct him he would do things his way and he would have backing from other people.

Were you typically aware of what other team members were working on?

- Generally speaking, but not. Particularly not <usability engineer>. I pretty much knew what <developer> was working on. I would talk to <qa person> occasionally. I thought testing was pretty on track.

How did you communicate with other team members during the project (e.g. email, walk to cubicle, phone call, IM)

- Face to face and I would send out emails to everyone about some things. I IMed <developer> a little bit.

Did you ever have a disagreement with another team member(s)? How was this resolved?

- <product manager>---he has a lot of other things to work on.
- <developer>---I've talked to him about it. That's his personality. He thinks he knows what should be done and he'll push for it.

Do you feel you received the support you needed from other team members to carry out your role in the project?

- In some ways. <developer> helped me with scheduling stuff, but it's one developer. There's no discussion about things. He also when we were disagreeing about things, he would give me estimates which were too big because he didn't think we should do it or too small b/c he thought we should do it.

How well overall do you feel the team collaborated? With respect to other projects you've been on?

- Even with all the problems, the product turned out nice. I think it turned out well considering all the above. All the personality issues, people all being new.

Usability

How well did the usability engineer work within the team?

- In some ways really well. In other ways, not so well. There was a perception that she was here just for the research. And some people thought it was just a waste of time. The perception that she was your sidekick didn't help either. But I think that everyone thinks that the interface is really good. They really like it. What amazes me is that—I really like people are really good at things that I don't have a good skill at—she came up with out of her head the interface—which is cool, and people like it. I think she and <developer> butted heads. (see above)

Is there value in having a separate usability engineer role on the team? What did the usability engineer contribute to the team?

- Yes, and you've seen matrices of the fact that some roles can overlap. I think that you have to have a distinct usability person. Distinct from the team lead, definitely not the developer.

Maybe overlap with documentation. Testing maybe. I think that for this project, it was really crucial. And it was important to have a separate distinct usability person. But I would not say that for all software projects. Especially since the goals were aligned such that usability was really, really important. Our initial requirements---correctly entering data, acceptance by user and speed---those all are directly affected by usability and that's not necessarily the case for all software projects.

Did you understand the sorts of things that the usability engineer did and why?

- No. At the time I understood very, very little. As things went along I learned more but I still don't know—I still don't know all the documents stuff the usability person could produce. I think that's something you and I should've talked about a whole lot more. I think it wasn't clear to me, why some things would be important and I had no say in what was important. And sometimes <usability engineer> would talk a lot about things that weren't important. And I think her job would've been a whole lot more easy if we had clarified where she should've been spending more of her time and what she should've shared with other people. It sounded like she was justifying what she was doing when she'd already done it. To do her job, she needed to connect the usability to her design decisions and that's good. But I don't think that sharing all that information with everyone was necessary.

What was your relationship with the usability engineer?

- I would say infrequently. Much less frequently than <developer>. And make sure if there things we needed to talk about, I knew to make time for that in meetings.

How well did you communicate with the usability engineer?

- I think I should've talked to her more and part of that due to her lack of experience. I think that someone with lots of experience, the amt that we talked would've been fine. We weren't on the same page sometimes (we were a lot, but we could've been more).

How important were usability evaluations in the conduct of the project?

- Sometimes I thought they were crucial, and other times it seemed like, --it goes back to the same thing—if you test things. It seemed like she wanted data to support her hypothesis as opposed to here is 'I tested this with six people and I asked them about the size'. It seemed like she was looking for scientific-specific data when in a real live project you don't need that detail of support for your opinion.
- Sometimes they were very important, some of them were really useful. Other ones, it was OK. Overall, they were useful. Some of them were really, really useful. Overall, I think it was worth it. But we're not collecting scientific data for publishing in a paper.

How does this compare to other projects you've been involved with?

- I've never been on a project that had a usability person.

What benefits/problems were encountered by incrementally developing the user interface? How does this compare with past projects?

- This project was perfect to use an agile process for b/c we didn't understand the problem. And until the very end we're still trying to figure out the problem. I don't think that's true for all software. I planned it in a really good way to do the time confirmation—and we worked out

how we wanted to do things in general on a smaller piece.—If we had done history first it would've been a larger more convoluted piece.

- What didn't happen was when the time confirmation should've been done, we kept revisiting things and we would go back. It wasn't as isolated as it should've been within an iteration. It is b/c the customer found things and it was b/c we were using an agile process and the customer was involved. If the customer hadn't been involved, we would've produced what we produced and then in version 2 fixed things. To just say it's a benefit is a lie. The major difficulty is, we have repeatedly gone back and retested the whole thing and this is a teeny, tiny project. If you had to do this for a big project, this is not a good thing. This is not a reasonable dev approach. And documentation, we just threw our arms up at the beginning and said we're not going to do this. But for a real product, we should've locked it down and not kept fixing it. So the constant customer involvement is a great thing, but to use management speak it adds risk. It inherently adds risk b/c the customer will tell you his opinion and if you don't do it you're doing a bad job and if you do do it then you're messing things up and you're not getting things done effectively.

Did incremental develop affect the overall quality of the user interface? Compared to designing the UI up-front?

What was the role of the usability tests with regard to the overall product? What did they contribute if anything? How does this compare to past projects?

Did you believe that usability design decisions were made with respect to high level goals of the project? Were any changes not aligned with high level goals?

- I think the vast majority of design decisions are opinion. And it seems like a lot of things are fairly---they don't matter that much. But I think that there were reasons and good reasons and sound reasons for the decisions that we focused on. But a lot of the design was more artistic, and I know usability people hate to hear it, they disagree, but...placement of buttons and things on the screen--does it really matter that much? Maybe.

Were there conflicts between team members with respect to which usability changes that need to be made? How were these resolved? (tradeoffs?)

- answer

What was the effect of having usability and development working in parallel? (generally having the usability engineer work 1 iteration ahead of the developer) What were the benefits? What were the problems?

- Here's one of the places where you and our places fell apart. When <usability engineer> and I sat down and came up with how things would fit and be done. And in hindsight, it seemed odd that you and I didn't sit down and do that. How it fit into development, I don't think it's as cut and dry as it would seem. Because I took it and it looked good, but in fact it just doesn't work that way. In an agile process, the agile process we used, we're under a really, really tight schedule and I don't think that had anything to do with an agile process. What initially looked

at an easy 2-3 month project, should've been a four month project. And with phase 2, it's a six month project. And management would've had no problem pushing out all the dates but b/c we needed to fit into the time schedule with school starting in September. In terms of fitting in with development, I think a second time it would work a whole lot more like it does on paper. In an agile process, what happens if during a usability study you say 'oh my god, we should've looked at this'. Part of the study pointed out another thing we should've looked at. I don't, in fitting with the usability person in the schedule. I don't think that writing formal documents—well she didn't write that many formal documents—it just seemed like it sometimes. If there were a fast effective way of editing the CDR and presenting that to the entire team and letting them know the significance of it, it might facilitate things like that so the schedule crunch would not be such an issue. But other things, people just don't care about—some software projects the user doesn't care about the user interface. I generally don't care. I am personally very concerned about functionality.

Were there conflicts between the UI design and development constraints? How were these resolved?

- See above

Information sharing/Project management

How often did you refer to usability documentation on the portal? How was it used?

- I went to the usability link frequently but a lot of that was to look at the <customer company>, customer artifacts. The only time I looked at those (usability tests) when after she talked about them and I glanced at them. The details---I hardly referred to them at all.
- Yes, I relied on her communicating the information that I needed to know. Once she told me the information, I did not have a whole lot of use to refer back to it.

Goals?

Scenarios?

Mockups?

Claims/design tradeoffs?

Iteration plan?

How did you share information with the team?

- At meetings and via email. Mostly at meetings. And one and one meetings (walk up and talk to someone) That's my preferred mode of communication. Which is not such a great thing around here.

Did you encounter any problems sharing information with the team or getting information related to the project?

- **Yes. Mostly due to time constraints. The only problems I had with that kind of stuff was me not being here enough and <usability engineer> not being here enough. And I thk it introduced stress points that ---if the only time someone is talking to you is to resolve issues—it's not good. You should have other times where you do communicate and...**

- **Time constraints.**

How aware were you of development progress throughout the project? Were you ever surprised by changes made to the system that you didn't expect?

- I remember once or maybe twice, that something was done a little faster than I was expecting it. So in terms of the schedule, things started up incredibly slowly and things didn't get going like I wanted them to. That was another example of <developer> going off and doing something. We weren't going to use <web technology> and then he went off—spent three days and then convinced <management> that that's what we should do and we got started a week late.

How did you use Team System if at all during the project? Did you notice any problems with how it was being used?

How important were the different types of meetings held during the project (daily, feature team, review meetings with customer)?

- I thought the daily meetings were really important to get the day going, especially Monday morning. I think a morning meeting—I don't know that it's necessarily every day but I thought it was good when I was here mon, wed, fri. And the customer meeting. Actually, if we could've met with him twice a week for an hour that would've been [preferred]. He was a whole lot more involved than a customer typically is in most development by far. And I think one thing that could be better defined is how you do want to interact with the customer. The customer doesn't care about the buttons and widgets in <web technology>. They don't care about if statements in your code so you can't have those types of discussions. You need to figure what things you need to discuss with the customer. (But are they, to take that a step farther, in a different direction, I don't know if the end user should be bothered with the details of the usability studies but I'm not sure)

How well did the team stay on schedule during the project?

- See above (and below)

How would you compare project velocity to other projects you have been involved in?

- After things got going, things were delayed a bit here and there and that's pretty typical. Like doing a customer site visit—we still haven't done that. I thought at the least we would go like—when phase 1 was done---we would do the trip and it never happened. Here's why?: I thought we could go up and do a demo for them with the UI, and somehow get a copy of their data like some tables from their <BMS> database so we could just show their real live data on our system. You can't do any of that. So the alternative is go up there and put our software on their system connecting to their test <BMS> system and to do that you have to get their <BMS> IT people to approve it. Now I understand all that, but 6 weeks ago I was still saying when are we going.

What things were done to address problems with project velocity?

How often did requirements change throughout the course of the project? How were they handled?

How does this compare to past projects you've been involved in?

- see above

Others

What are your thoughts on the development process used for this project?

That's the general sentiment among developers. I think you and I could've better distinguished that at the beginning and more distinctly. And what was the research aspect of this. If we were just doing an agile process, then you don't have to have a usability person so what was the real research part of this that we were trying to integrate the usability into the team and process. The research aspect was not clear at all.

How does it compare to other projects you've been on? If <usability engineer> had been a meridium employee, it would've been tremendously different. Cause the perception was she was just here as a part of the research. Hell, if I had been up here 5 days a week, it would've helped.

What things would you have done differently?

Any other problems/issues you encountered?

Appendix D. IRB informed consent and approval

Informed Consent

Informed Consent for Participant of Investigative Project Virginia Polytechnic Institute and State University and Meridium, Inc.

Title of Project: STTR Phase I: Integrating scenario-based usability engineering and agile software development practices

Investigators: D. Scott McCrickard, Jason Chong Lee

I. THE PURPOSE OF THIS RESEARCH/PROJECT

You are invited to participate in a study evaluating the use of Extreme Scenario-based Design (XSBD), a new agile design methodology that incorporates usability practices from scenario-based design. The study involves gathering a team of developers to use the approach in developing a novel Meridium application. This is a joint research effort between Meridium, Inc. and Virginia Tech.

II. PROCEDURES

You will be a member of a development team made up of Meridium employees that will develop a novel handheld application for a potential client. This development project will start and finish in the first half of 2008. The exact starting date will depend on the schedules of the development team but the project is expected to take approximately four months.

The development project will follow 2-week iterations. At the end of each iteration, some portion of the software application will be completed (coded and tested). At the end of each iteration, you will complete a questionnaire relating to the efficacy of the XSBD process. These questionnaires, which will include a mix of Likert-scale questions and written responses will gauge how well team members are interacting with each other (with respect to the process), and how the use of the approach in general is helping and hindering development progress. The questionnaire will be short and will be completed in 30-60 minutes.

You will also participate in a project retrospective at the end of every other iteration. In each project retrospective, the entire team will meet together to discuss and reflect on the design process. These retrospectives are intended to give you an opportunity to learn about the agile processes and models in use and provide feedback to improve them throughout the project. Retrospectives will generally involve looking at what did and did not work over the past iterations, what the underlying causes of problems are, and what will be done to address problems and maintain successes. Notes will be taken during each retrospective to capture relevant results. Each retrospective will last between 60-90 minutes.

All collected data from the study including those from questionnaires, retrospectives and metrics data will be anonymized such that you will not be tied to the data. In addition, data will be anonymized such that proprietary information related to Meridium products is not revealed.

III. RISKS

Participation will involve working at a workstation or with a laptop and will include regular meetings at the Meridium offices. The physical and mental aspects of the experiment are not designed to be stressful in any way and will not be significantly different from standard development practices at Meridium. If you become uncomfortable at any point in the study, you will be allowed to leave the project with no penalty.

IV. BENEFITS OF THIS PROJECT

Your participation in this project will provide information that may be used to improve the extreme scenario-based design process. No guarantee of benefits has been made to encourage you to participate by the researchers beyond your existing salary/benefits as an employee of Meridium. You may receive a synopsis summarizing this research when completed. Please leave an email address with the experimenter and a copy of the results will be sent to you.

VT IRB – This document is valid from 12 October 2007 – 11 October 2008

V. EXTENT OF ANONYMITY AND CONFIDENTIALITY

The results of this study will be kept strictly confidential. Your written consent is required for the researchers to release any data identified with you as an individual to anyone other than personnel working on the project. The information you provide will have your name removed and only a subject number will identify you during analyses and any written reports of the research.

VI. COMPENSATION

Your participation is voluntary and you will receive no compensation beyond your existing salary/benefits as an employee of Meridium.

VII. FREEDOM TO WITHDRAW

You are free to withdraw from this study at any time for any reason.

VIII. APPROVAL OF RESEARCH

This research has been approved, as required, by the Institutional Review Board for projects involving human subjects at Virginia Tech, as agreed upon with an Individual Investigator agreement between Meridium, Inc. and Virginia Tech.

IX. SUBJECT'S RESPONSIBILITIES AND PERMISSION

I voluntarily agree to participate in this study, and I know of no reason I cannot participate. I have read and understand the informed consent and conditions of this project. I have had all my questions answered. I hereby acknowledge the above and give my voluntary consent for participation in this project. If I participate, I may withdraw at any time without penalty. I agree to abide by the rules of this project

Signature

Date

Name (please print)

Contact: phone or address or

email address (OPTIONAL)

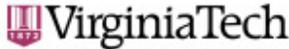
Should I have any questions about this research or its conduct, I may contact:

Investigators: D. Scott McCrickard
Associate Professor, Computer Science Department, Virginia Tech
Email: mccricks@cs.vt.edu
Phone: (540) 231-6698

Jason Chong Lee
Graduate Student, Computer Science Department, Virginia Tech
Email: chonglee@vt.edu

Review Board: David M. Moore, IRB Chair
Office of Research Compliance,
2000 Kraft Drive, Suite 2000 (0497), Blacksburg VA 24060
Email: moored@vt.edu
Phone: (540) 231-4991

IRB Approval Documents



Office of Research Compliance
Institutional Review Board
2000 Kraft Drive, Suite 2000 (0497)
Blacksburg, Virginia 24061
540/231-4991 Fax 540/231-0959
e-mail moored@vt.edu
www.irb.vt.edu

PWA00000572 (expires 1/20/2010)
IRB # is IRB00000867

DATE: October 12, 2007

MEMORANDUM

TO: Donald S. McCrickard
Jason Lee

Grant Compared 10/11/07

Approval date: 10/12/2007
Continuing Review Due Date: 9/27/2008
Expiration Date: 10/11/2008

FROM: David M. Moore 

SUBJECT: **IRB Expedited Approval:** "STTR Phase I: Integrating Scenario-Based Usability Engineering and Agile Software Development Practices", IRB # 07-496

This memo is regarding the above-mentioned protocol. The proposed research is eligible for expedited review according to the specifications authorized by 45 CFR 46.110 and 21 CFR 56.110. As Chair of the Virginia Tech Institutional Review Board, I have granted approval to the study for a period of 12 months, effective October 12, 2007.

As an investigator of human subjects, your responsibilities include the following:

1. Report promptly proposed changes in previously approved human subject research activities to the IRB, including changes to your study forms, procedures and investigators, regardless of how minor. The proposed changes must not be initiated without IRB review and approval, except where necessary to eliminate apparent immediate hazards to the subjects.
2. Report promptly to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.
3. Report promptly to the IRB of the study's closing (i.e., data collecting and data analysis complete at Virginia Tech). If the study is to continue past the expiration date (listed above), investigators must submit a request for continuing review prior to the continuing review due date (listed above). It is the researcher's responsibility to obtain re-approval from the IRB before the study's expiration date.
4. If re-approval is not obtained (unless the study has been reported to the IRB as closed) prior to the expiration date, all activities involving human subjects and data analysis must cease immediately, except where necessary to eliminate apparent immediate hazards to the subjects.

Important:

If you are conducting **federally funded non-exempt research**, this approval letter must state that the IRB has compared the OSP grant application and IRB application and found the documents to be consistent. Otherwise, this approval letter is invalid for OSP to release funds. Visit our website at <http://www.irb.vt.edu/pages/newstudy.htm#OSP> for further information.

As indicated on the IRB application, this study is receiving federal funds. The approved IRB application has been compared to the OSP proposal listed above and found to be consistent. Funds involving procedures relating to human subjects may be released. Visit our website at www.irb.vt.edu for further information

cc: File

Invent the Future

VIRGINIA POLYTECHNIC INSTITUTE UNIVERSITY AND STATE UNIVERSITY

An equal opportunity, affirmative action institution

DATE: September 12, 2008

MEMORANDUM

TO: Donald S. McCrickard
Jason Lee

FROM: David M. Moore 

Approval date: 10/12/2008
Continuing Review Due Date: 9/27/2009
Expiration Date: 10/11/2009

SUBJECT: **IRB Expedited Continuation 1:** "STTR Phase I: Integrating Scenario-Based Usability Engineering and Agile Software Development Practices", IRB # 07-496

This memo is regarding the above referenced protocol which was previously granted expedited approval by the IRB. The proposed research is eligible for expedited review according to the specifications authorized by 45 CFR 46.110 and 21 CFR 56.110. Pursuant to your request, as Chair of the Virginia Tech Institutional Review Board, I have granted approval for extension of the study for a period of 12 months, effective as of October 12, 2008.

Approval of your research by the IRB provides the appropriate review as required by federal and state laws regarding human subject research. As an investigator of human subjects, your responsibilities include the following:

1. Report promptly proposed changes in previously approved human subject research activities to the IRB, including changes to your study forms, procedures and investigators, regardless of how minor. The proposed changes must not be initiated without IRB review and approval, except where necessary to eliminate apparent immediate hazards to the subjects.
2. Report promptly to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.
3. Report promptly to the IRB of the study's closing (i.e., data collecting and data analysis complete at Virginia Tech). If the study is to continue past the expiration date (listed above), investigators must submit a request for continuing review prior to the continuing review due date (listed above). It is the researcher's responsibility to obtain re-approval from the IRB before the study's expiration date.
4. If re-approval is not obtained (unless the study has been reported to the IRB as closed) prior to the expiration date, all activities involving human subjects and data analysis must cease immediately, except where necessary to eliminate apparent immediate hazards to the subjects.

As indicated on the IRB application, this study is receiving federal funds. The approved IRB application has been compared to the OSP proposal listed above and found to be consistent. Funds involving procedures relating to human subjects may be released. Visit our website at www.irb.vt.edu for further information

cc: File

Invent the Future

DATE: September 14, 2009

MEMORANDUM

TO: Donald S. McCrickard
Jason Lee
Eric Ragan

FROM: David M. Moore 

Approval date: 10/12/2009
Continuing Review Due Date: 9/27/2010
Expiration Date: 10/11/2010

SUBJECT: **IRB Expedited Continuation 2:** "STTR Phase I: Integrating Scenario-Based Usability Engineering and Agile Software Development Practices", IRB # 07-496

This memo is regarding the above referenced protocol which was previously granted expedited approval by the IRB. The proposed research is eligible for expedited review according to the specifications authorized by 45 CFR 46.110 and 21 CFR 56.110. Pursuant to your request, as Chair of the Virginia Tech Institutional Review Board, I have granted approval for extension of the study for a period of 12 months, effective as of October 12, 2009.

Approval of your research by the IRB provides the appropriate review as required by federal and state laws regarding human subject research. As an investigator of human subjects, your responsibilities include the following:

1. Report promptly proposed changes in previously approved human subject research activities to the IRB, including changes to your study forms, procedures and investigators, regardless of how minor. The proposed changes must not be initiated without IRB review and approval, except where necessary to eliminate apparent immediate hazards to the subjects.
2. Report promptly to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.
3. Report promptly to the IRB of the study's closing (i.e., data collecting and data analysis complete at Virginia Tech). If the study is to continue past the expiration date (listed above), investigators must submit a request for continuing review prior to the continuing review due date (listed above). It is the researcher's responsibility to obtain re-approval from the IRB before the study's expiration date.
4. If re-approval is not obtained (unless the study has been reported to the IRB as closed) prior to the expiration date, all activities involving human subjects and data analysis must cease immediately, except where necessary to eliminate apparent immediate hazards to the subjects.

As indicated on the IRB application, this study is receiving federal funds. The approved IRB application has been compared to the OSP proposal listed above and found to be consistent. Funds involving procedures relating to human subjects may be released. Visit our website at www.irb.vt.edu for further information

cc: File

Invent the Future