Evaluating eXtreme Scenario-based Design in a Distributed Agile Team

Jason Chong Lee
Meridium, Inc.
207 Bullitt Avenue SE
Roanoke, Virginia 24013
jlee@meridium.com

Tejinder K. Judge
Center for HCI
Department of Computer Science
Virginia Tech
Blacksburg VA 24061-0106
tkjudge@vt.edu

D. Scott McCrickard
Center for HCI
Department of Computer Science
Virginia Tech
Blacksburg VA 24061-0106
mccricks@cs.vt.edu

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.
ACM 978-1-4503-0268-5/11/05.

Abstract
Enterprise-level organizations, which often rely on distributed development teams, are increasingly interested in finding ways to adopt agile and usability-focused methods. Agile usability researchers at Virginia Tech have partnered with Meridium, Inc. to look at how eXtreme Scenario-based Design (XSBD), an agile usability approach developed at Virginia Tech, can be used in a distributed environment. We report on the use of this XSBD approach in a distributed team at Meridium and how it addresses the challenges of an integrated approach through streamlined usability and development practices and clearly defined communication and information sharing practices.

Keywords
Usability, agile, distributed development, extreme scenario-based design

ACM Classification Keywords
H.5.2 Information interfaces and presentation (e.g., HCI): User Interfaces – *Theory and methods*; D.2.2 SOFTWARE ENGINEERING: Design Tools and Techniques – *User Interfaces*

General Terms
Design, Human Factors, Theory

Introduction

Agile software development methodologies have emerged as an effective way to deal with many of the risks of software development such as changing requirements, slipping development schedules and cost overruns by using practices like incremental test driven development and having onsite customers. However, agile methods did not originally incorporate practices from usability engineering. As a result, agile teams would often develop systems that were difficult to use [21]. To address this shortcoming, practitioners and researchers have been exploring ways to integrate usability into agile software development methodologies. However, they do not focus on how to use these integrated approaches in a distributed environment. This is problematic as agile methods are being increasingly adopted by enterprise-level organizations which often rely on distributed development teams.

This work is the start of a collaboration between Meridium Inc., a software development and consulting company specializing in asset performance management, and Virginia Tech to evaluate how well an agile usability approach can operate within a distributed agile development environment. eXtreme Scenario-based Design (XSBD)—the agile usability approach used in this case study—is derived from leading and established approaches from both the usability and agile domains and been used in collocated development teams [5, 14, 15, 24, 25]. Through our collaboration, we have identified three key challenges that needed to be addressed: synchronizing distributed usability and development efforts, promoting communication between team members, and effectively supporting document and artifact sharing between physically separated team members. Our experiences show how a combination of streamlined usability and development practices centered on a shared design representation along with well defined communication and tool-supported information management practices can help address those challenges.

In this case study, we first summarize the XSBD approach and then report on its use in a project at Meridium by a team distributed between USA and India. We describe both the successes and challenges encountered with this project and the changes made during the development effort to mitigate those challenges. We detail lessons learned in our instantiation of distributed agile usability and highlight areas for future work.

Background

In this section, we describe current approaches to incorporating agile methods into distributed organizations. We then describe established agile usability approaches and some of the challenges of applying them to a distributed environment.

Distributed agile

Agile software engineering processes are widely used and provide a multitude of benefits to companies that practice them. However as discussed by [3, 8, 19, 23, 27, 28], agile processes cannot be used in distributed environments unless they address the issues of communication and information sharing.

Face-to-face communication is a vital part of agile methods and needs to be taken into account and planned in advance (e.g. through video conferencing) to ensure participation from distributed teams [27].

Nidiffer et al. [19] mention that with a lack of communication the project will not be able to thrive in a distributed environment. Along the same lines, Angioni et al. [3] and Whitworth et al. [28] claim that communication and group awareness is important to establish interpersonal relationships to gain trust among team members and to establish a common vision of the project. Ramesh et al. [23] encouraged constant communication using practices such as short daily meetings, and by using online chat and Short Message Service (SMS) to improve communication in distributed teams.

Documentation of requirements and design need to be created and updated regularly to ensure all team members maintain common ground with regards to the product and the shared vision [19, 27]. However this recommendation clashes with the agile value of preferring working software over documentation [6]. The challenge here lies in finding the right balance. Ramesh et al., observed that some companies maintained a product/process repository to facilitate knowledge sharing [23]. The repository helped teams "report issues, assign priorities, and track project status". The companies also supplemented informal communication with relevant documentation. This facilitated collaboration and ensured a record of changes was created.

Cohn et al. [8] recommend that companies transitioning to agile wait at least two to three months after the initiation of an agile process before adding a distributed team. Companies need to "resolve their political and cultural issues" before the team can succeed working in a distributed environment. However if this is not possible, they recommend bringing people together in the first week or two of the process to build trust and open the lines of communication.

Agile usability

Agile methods typically follow an incremental development cycle in which each iteration includes some requirements analysis, design, implementation and testing [5, 24]. Usability engineering methods often follow a more phased approach in which requirements analysis and design is largely done up-front before implementation begins [25]. To address this conflict, an increasingly popular approach is one where some requirements analysis and modeling is done up-front so both developers and usability engineers can then work in parallel. Subsequent user interface modeling, design and evaluation occur continuously throughout development [1, 2, 7, 10, 12, 17, 18, 22, 26]. The advantage of this approach is that time and resource-intensive usability activities can occur in parallel with system implementation. These integrated approaches, however, typically involve collocated teams and do not focus on the communication, collaboration and information sharing issues specific to a distributed agile usability team.

Key challenges

Based on the preceding discussion and the results of our own case study, we can highlight three key issues that need to be addressed for a distributed agile usability team to work effectively:

1. Enabling collaboration in an agile team for which usability and development are not collocated.

2. Supporting communication between non-collocated usability and development teams when their respective daily work schedules do not overlap.
3. Effectively sharing documentation and design artifacts between non-collocated usability and development teams.

eXtreme Scenario-based Design

This section summarizes the XSBD agile usability approach developed by Lee and McCrickard [14, 15, 16]. The XSBD process draws on concepts from usability engineering and agile software development [5, 6, 24, 25]. In the XSBD process, the same core steps of scenario-based design—requirements analysis, activity design, and information and interaction design—are followed but proceed in concert with software development. This parallel approach is common in agile usability methods [7, 12, 18]. This allows potentially time-consuming and non-interdependent processes to occur in parallel. The XSBD approach assumes that an experienced usability engineer is a member of team and is working with the developers to implement the system.

Central Design Record

The Central Design Record (CDR) is the main user experience design representation used in XSBD to help ensure that the user interface meets the needs of the end users and customer. The CDR is based on Norman's concept of a system image and was adapted

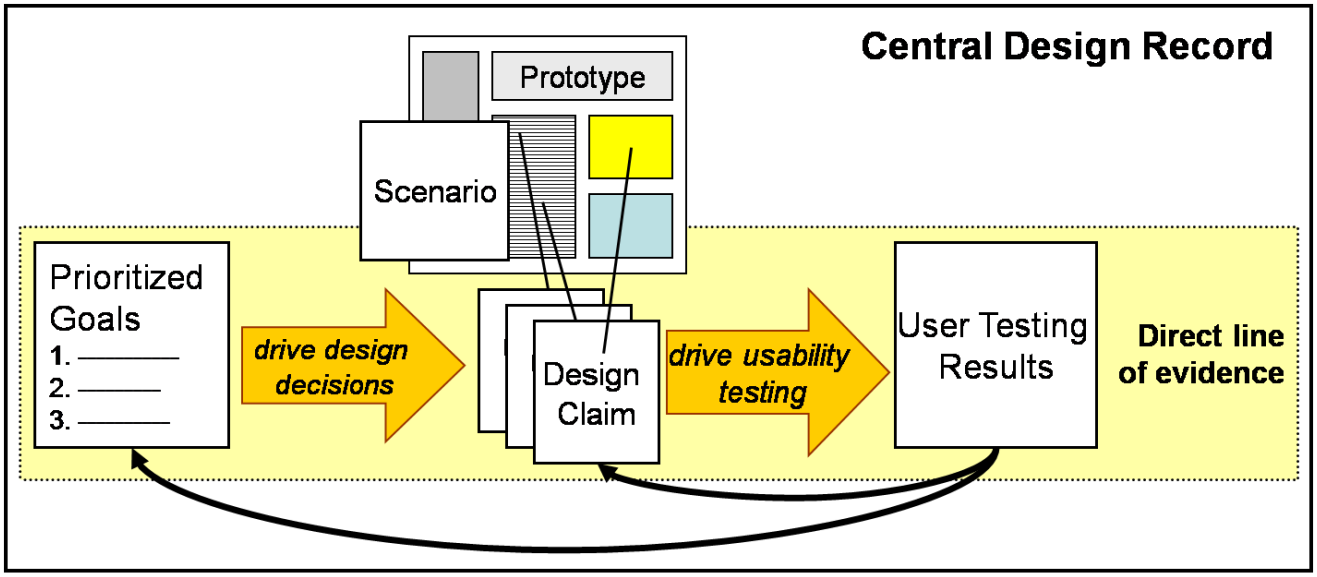

Figure 1. The CDR directly links prioritized goals to design decisions validated by user tests.

for use in scenario-based design by Lee and McCrickard [15, 16, 20]. 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 is primarily managed by the usability engineer with input from the rest of the team.

The CDR consists of three parts (See Figure 1):

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

Streamlined requirements documents including the prioritized project goals and vision statement help the entire team have an understanding of what system is being developed, for whom it is targeted and what aspects of it are most important. Within an incremental agile development framework, these goals are continuously revisited and revised as necessary.

Designs are captured using a mix of design prototypes, scenarios and claims to capture design decisions. These artifacts are generally developed within increments and are directly related to the features or stories (using XP terminology) that developers manage to schedule their development tasks. Design prototypes typically come in the form of mockups and are one of the primary mechanisms the usability engineer uses to communicate the UI design to developers. Scenarios—which describe common workflows in a narrative form—are used in conjunction with mockups to give developers an end-to-end understanding of how the system will be used [24]. Claims are feature descriptions with associated design tradeoffs that highlight the user ramifications of different aspects of the interface. They are similar to stories in that they are brief descriptions of system features but they are used by the usability engineer in XSBD to capture and guide design decisions rather than to aid in project planning and management.

Usability testing is used in XSBD to verify that design decisions have resulted in a usable system, validate that the design meets high level design goals, indentify new problems and uncover new requirements. It is analogous to testing practices common in agile approaches such as test-driven development and acceptance testing [5, 25]. In test-driven development, developers can create tests in code which can subsequently be used to verify the implemented functionality. These tests can be aggregated and automated. The usability analogue to these types of tests are in how claims are used. In XSBD, claims are linked to high level project goals. These in turn are verified through usability testing. These tests complete the chain linking designs goals to design decisions (see Figure 3). They allow the usability engineer to streamline the evaluation process (which unlike unit testing, is not easily automated) by focusing on only the most critical areas of the user interface [14, 15,

16]. This is important as evaluations are planned and run within a single development increment so results can be handed off to developers in the following iteration. Continually revisiting the high-level goals with each test cycle allows the usability engineer to run incremental tests while maintaining the overall vision of the design.

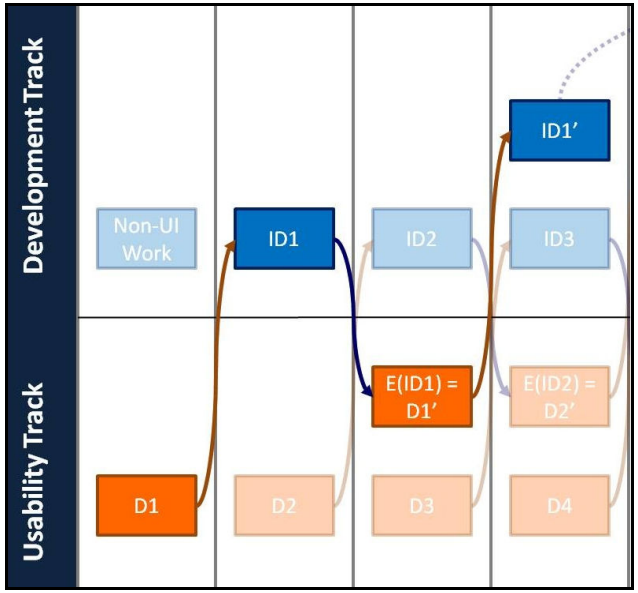

Figure 2. Synchronized usability and development tracks.

The XSBD Process

XSBD has two separate but synchronized usability and development tracks. The usability engineer works one iteration ahead of the developers so designs can be delivered for developers to start implementing those designs in the following iteration. These parallel tracks allow the team to optimize its velocity while still developing a system that meets high-level design goals. This type of parallel development process is common in other implementations of integrated agile usability approaches [7, 10, 12, 18, 26].

Figure 2 shows a simplified view of how the handoffs between the usability and development tracks occur. 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'**). Using this handoff system, the two tracks can work in parallel without bottlenecking each other (e.g. The usability engineer can work on designing **D2** while the developers implement **D1**). 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 typically used [14, 15].

Each project generally starts with an Iteration 0, to define the vision and goals for the project. During this iteration, the usability engineer will conduct an abbreviated requirements analysis process to begin gathering information for the CDR. The usability engineers and developers will stay in sync through regular iteration planning and review meetings as is done in XP. They may also communicate on a day to day basis through ad hoc face-to-face meetings, emails or instant messages.

Distributed XSBD in practice

A distributed web development project at Meridium, Inc. was used as a test bed for the XSBD process. This team was tasked with developing a simplified, online version of Meridium's primary data collection and analysis tool.

The project manager, product manager, testing lead, development lead, documentation person and usability engineer, hereafter referred to as the onsite team, was located at their main office in Roanoke, VA. The usability engineer was only working part-time on the team. The core development and testing team, hereafter referred to as the offsite team, was composed of developers and testers in India. The onsite team had less experience with agile methods, as Meridium, Inc. had only recently started to transition to agile development methodologies. However, the offsite team was experienced in agile methods and used a variation of Scrum—an agile development methodology that focuses on project management practices [25]. The distributed groups stayed in contact through daily meetings that were held using phone conferencing and screen-sharing software. They also used a variety of asynchronous communication and information-sharing tools such as email and IM.

We took an action research approach to developing and evaluating the distributed agile usability process, starting with the XSBD and Scrum processes as a base. Action research involves iterative problem solving by reflecting on what did or did not work within each iteration and is ideal for the type of research-in-practice approach undertaken here [4]. This allowed us to incrementally improve and adapt the XSBD process for use in this distributed team. This was needed since XSBD was originally only developed for and used in collocated teams [14, 15, 16].

Table 1. Example critical incident.

Role:	Usability Engineer
Date of incident:	4/9/2008
Event description:	I wanted to look over the [latest] prototype but I found out that it has not been updated for over a week.
Effect of incident:	It negatively affected my work as I needed to review the prototype to make usability recommendations.

Critical incidents—descriptions of events that positively or negatively affected the team—were recorded to gather information throughout the development process [11]. These were collected by the researchers through observations during daily meetings, planning meetings and other regular team meetings. We also had project team members record their own critical incidents through an online reporting form [13]. Generally, critical incidents contained the following information: the role of the person writing the incident, the date of the incident, a description of the incident, an explanation of how this positively or negatively affected the team and the perceived severity of the incident. An example critical incident captured using the online reporting form is shown in Table 1. A total of 90 incidents were recorded over a period of three months. After the end of the project, the researcher reviewed and grouped the incidents. As is typical with self-reporting of critical incidents, people tended to record incidents that negatively affected the team. This was

not viewed as problematic as it allowed the researchers to identify and address problems as they arose during the project.

Table 2. Types of incidents captured during the case study.

Critical incident categories	**Occurrences**
Collaboration	21
Communication	21
Document & artifact sharing	35
Other	13

The three major groups of incidents are listed at the top of Table 2. The 'other' incidents primarily related to technical issues that were generally resolved quickly. In keeping with the action research model, the following sections will present incidents encountered during the project as they relate to the three major critical incident categories in the following format:

Issue: A description of the problem.

Action: The action taken to address the problem.

Analysis: An evaluation of how well the action addressed the issue.

We will present exemplar incidents from each of the major groups of incidents. These are not meant to be comprehensive but rather show what specific types of incidents occurred and how they were addressed.

Collaboration Issues

This section describes collaboration issues that were encountered—especially with respect to how developers and the usability engineer worked together. Many of the incidents centered on the fact that the development team was not accustomed to working with a usability engineer and because they tended to complete work faster than the usability engineer could keep up with. The problems were addressed by having the usability engineer assigning some of the design work to the developers while ensuring that the resulting designs met the high level project goals. An exemplar incident is described below.

Issue: Early on, the usability engineer had trouble working ahead of developers. The usability engineer was brought on board during the first iteration after the developers had already started developing the UI. This resulted in designs that did not meet high-level design goals. In addition, the offsite developers were reluctant to rework what they had already done. Developers would come up with designs without consideration for target users and usability goals. This may have continued because the usability engineer was working part-time on the team. She was not present at all daily meetings and hence was not able to immediately answer questions that developers and other team members had. The quality assurance manager noted that:

> "This has a negative impact because it causes lots of confusion and dev and testing rework."

Action: Use the CDR concept of linking goals to evaluated design features to focus usability work only on the most important features of the system. The usability engineer had to first evaluate the existing implemented UI. She then quickly defined design goals, user description, claims and other aspects of the CDR as she worked to design for the next iteration to get ahead of the developers. The usability engineer focused on only the most critical features of the system

as defined by the goals. She generated claims to capture key tradeoffs of specific features that directly related to the high level goals of the system.

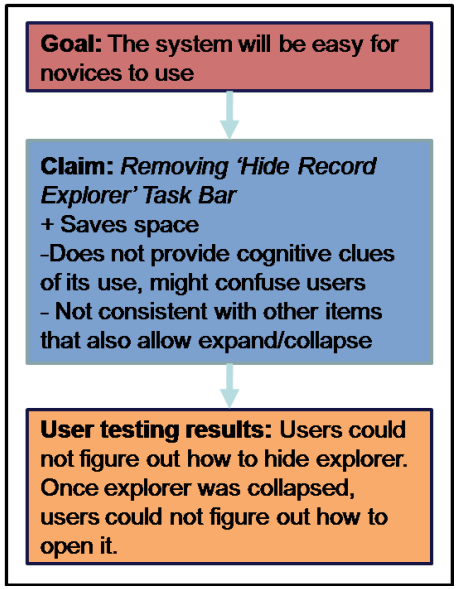

Figure 3. Example claim that is linked to a high level goal and is later verified through usability testing.

Figure 3 shows how a specific feature was linked to a high level goal and is later validated through usability testing. This feature related to how records were displayed in the web application and was one of the most important features in the system. In this case, there was some disagreement within the team about how the feature should be designed. After running a usability test, the usability engineer determined that the downsides outweighed the potential upside of saving space on the screen and redesigned the feature. Lightweight evaluations such as this were typically run in a single day with a small group of participants at Meridium who had similar characteristics with the end users of the system. Non-critical features were typically designed and implemented by the offsite developers and signed off on by the usability engineer after she reviewed them at the iteration review meetings.

Analysis: By using the CDR to focus on only the most important features of the system, the usability engineer was able to sync up with the developers. The usability engineer was able to catch up to developers by the third iteration and begin delivering feature designs and redesign requests to them to implement in the subsequent iteration. By having the offsite developers handle some of the design work, the usability engineer had more time to focus on the most important features of the system as defined by the high level goals in the CDR. In addition, over the course of several iterations, this gave the offsite developers a better understanding of what design aspects of the system mattered most. As shown in the example, being able to share and communicate the reasoning behind decisions based on goals, design claims and testing results helped the usability engineer better show the rest of the team why they were being made and why they were important.

Communication Issues

In this section we describe examples of communication problems that occurred and how they were addressed. Many of the communication problems were exacerbated by the fact that the onsite and offsite teams were separated by such a large distance. In addition, the fact that neither the usability engineer nor the offsite developers were accustomed to working with each other made it difficult for them to communicate

effectively early on in the project. Below, we present two illustrative incidents and how they were addressed.

Issue: At the beginning of the project there was some misunderstanding regarding the role of the usability engineer. The development lead and other members of development seemed to think that the usability engineer just designed UI mockups. They did not understand other things a usability engineer did in terms of user tasks analysis, usability evaluations, etc. Hence the development lead wanted the usability engineer to come up with specific ideas for UI in the first iteration, without doing any user task analysis. There was a disconnect between what the development lead expected and what the usability engineer wanted to do. This problem arose due to the project team not having the experience of working with a usability engineer. The product manager later noted the importance of the usability engineer to the team and the problem associated with having team members not understanding her role:

> "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…"

Action: The development team held an 'official kickoff' meeting to introduce the team members to one another. The usability engineer and documentation person joined the team after development had already begun. This was because they were not added to the team initially due to resource and scheduling constraints within Meridium. As a result, this kickoff meeting was held near the end of the first iteration. This kickoff meeting was also used as a way to introduce the usability engineer to the rest of the team and communicate what her role and responsibilities would be.

Analysis: The kickoff meeting was not helpful in helping team members understand the role of the usability engineer. The meeting was focused more on introducing the system itself and could only briefly touch on the usability engineer. As Cohn suggested, it would have been better to bring the entire team together at the start of the project to open lines of communication [8] early on—although it is not clear how much this would have helped in making the team understand the role of the usability engineer. In the end, the best way to get the team to understand the role and value of the usability engineer was for the usability engineer to present work products such as mockups and user testing results to the rest of team—essentially to demonstrate her value through actions. The streamlined XSBD development process helped in that the usability engineer was able to work relatively quickly and deliver designs and user feedback to the developers in a timely fashion.

Issue: The usability engineer and developers were not communicating effectively. Early on, there were times where developers did not fully understand a UI mockup but took no initiative to resolve it by contacting the usability engineer. This may have been because the offsite developers were not used to working with a usability engineer. The offsite developers wanted to maintain velocity and did not want to wait a day to receive feedback from the usability engineer (see Figure 4).

Action: Actively leverage asynchronous communication technologies. To address this communication issue, the usability engineer repeatedly reminded the team to email her with questions when she was not available. The usability engineer also proactively initiated communications through emails and made sure to promptly follow up on issues identified during daily meetings she did not attend.

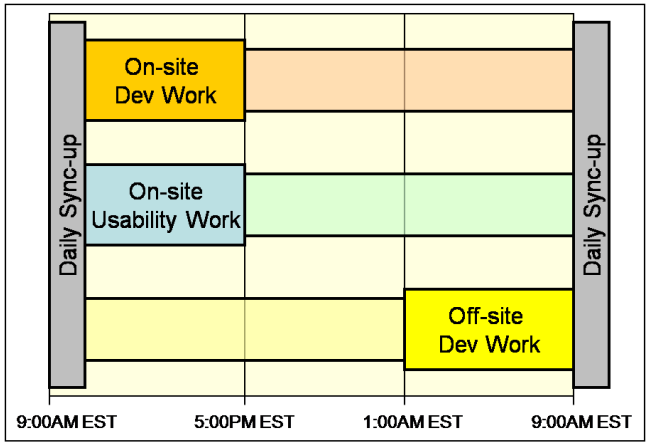

Figure 4. The 24-hour development cycle divided into 8 hour slices. Onsite and offsite work occur at different times during the same day. Daily meetings are used to stay synchronized.

Analysis: Increased communication between the usability engineer and the rest of the team. As the project progressed, the entire team increased its appreciation of having the usability engineer and communication became easier and more frequent. This was important in that it allowed the usability engineer to establish and maintain a working relationship with the offsite team and to clear up misunderstandings and differences between her designs and the implementations. Later on, the team began using an online discussion board as a way to provide a central location through which issues could be asynchronously identified, tracked and addressed. This was an improvement over emails as issues could get lost in email inboxes and were sometimes hard to locate.

Documentation and Artifact Sharing Issues

With the team being distributed, sharing up-to-date documents and design artifacts and maintaining them proved to be problematic, especially since the team was unable to communicate synchronously most of the time. These problems were addressed by improving communication between the separated groups as described in the previous section and also be introducing a more structured, regimented information sharing process then one might find in a collocated agile team. We present the following incident to illustrate how this was done.

Issue: Team had problems sharing and using up-to-date information. At an early daily meeting, the problem of sharing design documents and information between distributed team members was brought up. The project team used email to share documents resulting in the problem of documents getting out of sync. For example, the usability engineer might send out a version of the design for the upcoming iteration and then later make a slight change to it and send it out again. One of the offsite developers might accidentally implement the older design. The distributed nature of the team magnified these issues as the team could not communicate synchronously and the problem might not surface until the iteration review meeting. In addition, if the usability engineer or other

team members missed a meeting it was difficult to get synced up. This was somewhat common as the daily meetings occurred near the beginning of the day for the onsite members and at the end of the day for the offsite members. As one developer noted:

> "There is no way for the developers to convey the details about what is being delivered… 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."

Action: Use structured collaboration process centered on an online team portal to share information. The project manager mentioned that he would like to start using the Microsoft SharePoint portal more as a shared workspace since the team is distributed. The portal was used to store announcements, discussions, agendas and other work documents including usability design documents that were part of the CDR. This made it easier for the offsite and onsite teams to share and disseminate project information.

The team also clearly defined the mechanisms through which the usability engineer would deliver designs and how the team would communicate design issues as they arose. The usability engineer would meet at the beginning of each iteration with the product manager to validate the design to be implemented in the next iteration. The usability engineer then uploaded that design to the online team portal by the end of the day. When possible, the offshore developers would demo implemented functionality during the daily meetings so the usability engineer could compare against the designs and also so any small issues the developers had with the design could be addressed. Any other questions or issues that the developers encountered were addressed by posting those issues in the online discussion board. The issues were then reviewed at the start of each daily meeting. 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, she would document them in the form of claims. These claims would then either be tested later by the usability engineer or otherwise resolved through by consensus opinion of the team.

Analysis: Improved information sharing within the team at the cost of increased documentation and process structure. The structured process and tools the team used to collaborate and share information may seem counter to the agile principle of valuing "individuals and interactions over processes and tools" [14] at first glance. Although the team had to document more information and share it in a more structured way, it improved the way that the distributed team members were interacting and communicating with each other. This compromise minimized the amount of additional documentation and allowed the offsite development team to make quick fixes when needed. It also reduced confusions and misunderstandings, as the team was previously not using a unified way to track such changes to the system.

Lessons learned

During this effort, we integrated the XSBD agile usability design process into a distributed development team. In practice, we found that many of the challenges the team encountered were related to communication, collaboration and information sharing

between team members that were onsite versus offsite as well as between usability engineers and other members of the development team. Overall, we found that the adoption of the XSBD approach was hampered by the distributed nature of the team since the effective use of the approach depends on consistent and regular communication between team members. In particular, the team did not adopt and use claims as extensively as when the XSBD approach was used by a collocated team [14]. However, the usability engineer found the CDR useful in helping her focus on only the most critical areas of the interface and in making sure high level goals were being met. In summary, we can recommend the following:

- Have the usability engineer focus on only the most important aspects of the interface so she can stay ahead of and synchronized with development. This was done here by directly linking goals, design decisions and usability tests through the CDR. Other team members can design and develop noncritical areas of the UI that can be validated later by the usability engineer.
- Define high level goals—accomplished in this project through the CDR—that are shared and agreed upon by all team members—at the beginning of the project and continuously verify that they are being met. This was done by the usability engineer through user testing. This allowed the usability engineer to both communicate design decisions and their importance to other team members and to demonstrate her value to the rest of the team.
- Clearly define roles for all team members on the project. Not doing this initially hampered our development effort as many of the developers were not accustomed to having a usability engineer on the team. This can be done by having an initial iteration where team members are brought together to open lines of communication early on as suggested by Cohn et. al [8].
- Clearly define communication and information sharing mechanisms to allow non-collocated team members to remain in sync. Expect to document and track more information than in a collocated agile team. Non-synchronous communication mechanisms like an online portal and discussion boards were important as developers and the usability engineer often could not communicate synchronously.

Future work

This work is part of an ongoing collaboration between Virginia Tech and Meridium. We are continuing to evaluate the use of the XSBD approach in several development projects. We will continue to explore how software developers and usability engineers can interact most effectively in agile teams using the CDR—and whether it can be used to better support information sharing between distributed team members. This will allow us to continue to improve and adapt the XSBD approach, collect data from teams over longer periods of time, and form teams that are more experienced with XSBD and agile methods in general.

In addition, we are looking at broader communication and collaboration issues related to interactions between usability engineers other team members. Specifically, we plan to study how quality assurance and documentation personnel can work within the XSBD process. In addition, as XSBD is more widely adopted

within Meridium, we will need to look at how it impacts project management issues such as long-term release planning and resource allocation. This will closely relate to the issue of identifying what sorts of projects and teams the XSBD process is best suited towards.

Acknowledgements

Thanks to the development team at Meridium for their help and insights during the project. This material is based upon work supported by a National Science Foundation Small Business Technology Transfer Phase I Grant (#0740827).

Works Cited

[1] Ambler, S. W. Introduction to Agile Usability: User Experience Activities on Agile Development Projects. Ambysoft, Inc., 2007. www.agilemodeling.com/essays/agileUsability.htm

[2] Ambler, S. W., and Jeffries, R. Agile Modeling: Effective Practices for Extreme Programming and the Unified Process. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[3] Angioni, M., Sanna, R., and Soro, A. Defining a Distributed Agile Methodology for an Open Source Scenario. In Proc. OSS 2005, (2005), 209-214.

[4] Avison, D., Lau, F., Myers, M. and Nielsen, P. A. Action Research. *Communications of the ACM 42,* 1 (1999), 94-97.

[5] Beck, K., and Andrews, C. Extreme Programming Explained: Embrace Change, 2nd Edition. Addison Wesley Professional, Boston, MA, USA, 2004.

[6] 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. The Agile Manifesto. 2001. http://agilemanifesto.org

[7] Chamberlain, S., Sharp, H., and Maiden, N. Towards a Framework for Integrating Agile Development and User-Centred Design. In Proc. XP 2006, Springer Berlin/Heidelberg (2006), 143-153.

[8] Cohn, M., and Ford, D. Introducing an Agile Process to an Organization. Computer 36, 6 (2003), 74-78.

[9] Constantine, L. L., Process Agility and Software Usability: Toward Lightweight Usage-Centered Design. Information Age 8, 2 (2002). Reprinted in L. Constantine (Ed.), Beyond Chaos: The Expert Edge in Managing Software Development. Addison-Wesley, Boston, MA, USA 2001.

[10] Ferreira, J., Noble, J., and Biddle, R. Agile Development Iterations and UI Design. In Proc. Agile 2007, IEEE Computer Society (2009), 50-58.

[11] Flanagan, J. C. The Critical Incident Technique. Psychological Bulletin 51, 4 (1954), 327-358.

[12] Fox, D., Sillito, J., and Maurer, F. Agile Methods and User-Centered Design: How These Two Methodologies are Being Successfully Integrated in Industry. In Proc. Agile 2008, IEEE Computer Society (2008), 63-72.

[13] Hartson, H. R., Castillo J. C., Kelso, J., and Neale, W. C. Remote evaluation: the network as an extension of the usability laboratory. In Proc. CHI 1996, ACM Press (1996), 228-235.

[14] Lee, J. C., Stevens, K. T., and McCrickard, D. S. Examining the foundations of agile usability with eXtreme Scenario-based Design. In Proc. Agile 2009, IEEE Computer Society (2009), 3-10.

[15] Lee, J. C. and McCrickard, D. S. Towards Extreme(ly) Usable Software: Exploring Tensions Between Usability and Agile Software Development. In Proc. Agile 2007, IEEE Computer Society (2007), 59-71.

[16] Lee, J. C., Wahid, S., McCrickard, D. S., Chewar, C. M., and Congleton, B. Understanding Usability: Investigating an Integrated Design Environment and

Management System. International Journal of Information Technology and Smart Education (ITSE) 4, 3 (2007), 161-175.

[17] Meszaros, G. and Aston, J. Adding Usability Testing to an Agile Project. In Proc. Agile 2006, IEEE Computer Society (2007), 23-28.

[18] Miller, L. Case Study of Customer Input For a Successful Product, In Proc. Agile 2005, IEEE Computer Society (2005), 225-234.

[19] Nidiffer, K., and Dolan, D. Evolving Distributed Project Management. IEEE Software 22, 5 (2005), 63-72.

[20] Norman, D. A. Cognitive engineering. In Norman, D. A., and Draper, S. W. (Eds.) User Centered Systems Design: New Perspetives on Human-Computer Interaction. Lawrence Erlbaum Associates, New Jersey, USA, 1986.

[21] Patton, J. Hitting the target: adding interaction design to agile software development. In *Proc. OOPSLA 2002*, ACM Press (2002), 1-ff.

[22] Poppendieck, T. The Agile Customer's Toolkit. Poppendieck LLC., 2003. http://www.poppendieck.com/pdfs/Agile_Customers_Toolkit_Paper.pdf

[23] Ramesh, B., Cao, L., Mohan, K., and Xu, P. Can Distributed Software Development Be Agile? Communications of the ACM 49, 10 (2006), 41-46.

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

[25] Schwaber, K., and Beedle, M. Agile Software Development with SCRUM. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[26] Sy, Desiree. Adapting Usability Investigations for Agile User-Centered Design. Journal of Usability Studies. 2(3), 112-132.

[27] Turk, D., France, R., and Rumpe, B. Limitations of Agile Software Processes. In Proc. XP 2002, Springer-Verlag (2002), 43-46.

[28] Whitworth, E., and Biddle, R. The Social Nature of Agile Teams. In Proc. Agile 2007, IEEE Computer Society (2007), 26-36.