Towards Extreme(ly) Usable Software: Exploring Tensions Between Usability and Agile Software Development

Jason Chong Lee and D. Scott McCrickard
Center for Human-Computer Interaction, Department of Computer Science
Virginia Tech, Blacksburg, VA 24061-0106
{chonglee, mccricks}@cs.vt.edu

Abstract

Design is an inherently multidisciplinary endeavor. This raises the question of how to develop systems in ways that can best leverage the perspectives, practices, and knowledge bases of these different areas. Agile software development and usability engineering both address important aspects of system design, but there are tensions between the methods that make them difficult to integrate. This work presents a development approach that draws from extreme programming (XP), a widely practiced agile software development process, and scenario-based design (SBD), an established usability engineering process. It describes three key questions that need to be addressed for agile software development methods and usability engineering practices to work together effectively, and it introduces interface architectures and design representations that can address these questions.

1. Introduction

The growing importance of computing systems in everyone's daily lives has made software development an inherently multidisciplinary endeavor [17][18]. This raises the question of how to develop systems in ways that can best leverage the perspectives, practices and knowledge bases of these 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. The broad goal of this research is to address the problems associated with multidisciplinary system design—focusing specifically on agile software development and usability-—by developing a design process that supports the effective creation of usable software-based systems through common practices and toolsets derived from both areas. Initially, agile methods such as extreme programming provided little guidance on how to incorporate best practices from usability engineering [2]. The underlying assumption appeared to be that having an active, on-site customer would result in a usable end product. However, this turned out not to be the case. Systems could be developed that were functionally correct but still hard to use [19]. Usability can help by giving developers a more in-depth understanding of users, their work and how their goals can be realized most efficiently. This realization has led to a surge of interest in usability within the agile community [2][5][14][19][24].

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.

This paper will first present background information on usability and current work on integrating usability with agile methods. It will then summarize some of key questions that need to be addressed to mitigate the tensions between the two areas and present an overview of the combined XP+SBD process. The results of two design case studies and the following discussion will reflect on our approach and its benefits and limitations. Finally, we will discuss further improvements and implications for tool support.

2. Background and related work

This section provides some background on usability engineering practices and other work that is being done on integrating usability and agile software development. This background information will highlight some of the conflicting methodologies and practices of the two areas which serve as motivation for this work.

2.1 The need for usability

Usability engineering is concerned with developing interfaces that people can use efficiently and effectively. It deals with issues such as system learnability, efficiency, memorability, errors and user satisfaction [7][9][22]. Usability engineering processes are important in that they focus on developing systems that are tailored for end users. Its underlying practices and theories can give insights into user motivations, characteristics and work environments and draw on many different areas including psychology, sociology, physiology and human factors. Agile methods and usability practices have much in common. They both follow cyclical development cycles, are human-centered and both emphasize team coordination and communication. However, differences in the philosophies of the two areas may cause conflicts that can hinder the development process.

One established usability engineering approach is scenario-based design, a design-representation based process that uses *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 [4][22]. Claims provide compact, designer-digestible packets of knowledge ideal for use in time-critical design activities [25]. Like many usability engineering approaches, SBD begins with an in-depth requirements analysis process followed by an iterative development/evaluation cycle. These development cycles are typically longer than XP iterations, and focus more on requirements gathering and low-fidelity prototyping early in the development process. These design practices, though important in usability, are not a good fit in many agile practices which focus on continuous of working software and minimal up-front design work.

SBD design practices allow usability engineers to design an interaction architecture that supports the users' tasks in an efficient and organized manner. Usability evaluations are typically conducted with actual end-users and can involve walkthroughs, longitudinal studies of use or controlled lab-based studies. Conducting usability evaluations and doing the subsequent analysis of the data can be time-consuming—especially with respect to development cycles as short as those in XP. This raises the question of how these types of design practices can be streamlined to fit in an agile framework.

2.2 Agile software and usability

Agile practitioners have begun to explore ways of incorporating usability into agile methods [24]. Development processes from both areas such as XP and SBD share many of the same foundational concepts including iterative development and a focus on users and communication. However, a joint approach is difficult because agile methods, which are incremental and iterative in nature, do not support any kind of comprehensive overview of the entire interface architecture which is an important part of making consistent and usable interfaces. Constantine advocates a combined usability and agile software development process that begins with interface design and then continues with existing agile software development processes [6]. One potential problem with this approach is that the interface usability design process becomes a bottleneck in the overall development process and violates many of the accepted tenets of the agile development philosophy [15]. Other approaches suggest a methodology where software development and usability engineering proceed in parallel [2][5][14][19][21]. This appears to be the preferred approach although communication and careful coordination are vital as agile developers and usability specialists can have differing motivations, thought processes and goals.

2.3 Tensions between agility and usability

This work will contribute to these continuing efforts by exploring some of the tensions between agile software development and usability from the perspective of SBD and XP. By looking at how well different usability methods and techniques can be incorporated into agile methods we hope to gain additional insights into opportunities for mutual benefit and support. These tensions lead to three key questions that need to be addressed for agile software development methods and usability engineering practices to work together effectively.

1. How can developers design consistent and coherent interface architectures within an incremental agile development framework?
2. How can usability evaluations be streamlined so they better fit in accelerated development cycles while still providing useful results?
3. How can project members support communication and cooperation between designers, customers, users and other stakeholders who have different backgrounds and expertise?

The remainder of this work explores the questions above by exploring the practices in each approach that contribute to the tensions and by presenting a way that they can be mitigated.

3. XP+SBD process defined

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 XP+SBD process supports the best practices of both processes while mitigating the tensions between them. The key features of the process are defined below. Many XP practices which are similar to SBD are preserved including iterative development and release and iteration planning games. Others such as pair programming, unit testing, code refactoring and continuous integration are also unchanged.

3.1 Scenario-based design

Scenario-based design uses scenarios and claims to describe usage situations and highlight the tradeoffs of specific interface features. The scenario in Figure 1, based on the Notification Collage [8], describes the use of a virtual notice board to allow users to maintain awareness of people they work with.

Unlike the stories of XP, scenarios may involve many features of the system and will describe how one or more people engage in some activity [1][22]. They provide a realistic context of use from which to derive insights about the interface design. The claim in Figure 2 describes a specific design feature of the Notification Collage. Claims such as this can help designers and other stakeholders consider different design tradeoffs throughout the development process:

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 research-related information that is being shared within his lab. He uses the Notification Collage (NC), which runs on his second monitor, in order to be constantly aware of such information. Pascal can now casually 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 most recent 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.

Figure 1. Example scenario for the notification collage.

Information artifacts haphazardly posted in an unorganized fashion onto a public display for relevant information delivery, similar to how fliers are posted 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 although the relative age of an item that appears on top is newer, the actual age of an item is not apparent

Figure 2. Example claim describing tradeoffs of the bulletin board metaphor.

Scenario-based design specifies four design phases: requirements analysis, activity design, information design, and interaction design. *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, often intermingle in practice as design proceeds iteratively. For example, information and interaction design may occur in a single iteration, followed by a usability evaluation that prompts the designers to reconsider the overall activity design.

3.2 Interface architecture design

In the XP+SBD process, this same basic process is followed to develop the interface but it proceeds in concert with software development (Figure 3). Instead of an extended up-front requirements analysis phase, abbreviated requirements gathering activities such as stakeholder identification and task analysis will be conducted at the beginning of the project after client meetings. The root concept document is developed after an initial client meeting.

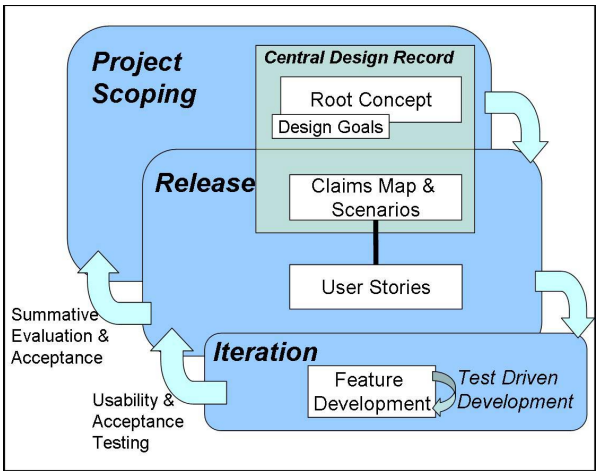

Figure 3. Key steps in XP+SBD process. Design artifacts are shown in white.

The interface design representation is called the *central design record* (CDR), and is part of our continuing research efforts [10][11]. It consists of the set of scenarios describing different usage situations, interrelated claims describing to 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 [16]. For example, critical parameters used during car design might be mileage or top speed. It is used to guide all stages of interface design in the XP+SBD 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 [27] (Figure 9). 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 XP+SBD 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 during each iteration.

3.3 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 [10][11]. 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.

3.4 Usability evaluations through the CDR

The other addition we make to the existing XP framework are 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 for end users. 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 to 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. Design case studies

Two design case studies are detailed below that demonstrate how the XP+SBD process addresses the key questions detailed in section 2.3. 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 XP+SBD 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 XP+SBD process outlined in Section 3 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 all points in the project [12]. All projects were developed using C# to run on PocketPCs. The NUnit[1] framework was used for unit testing in each project.

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.

4.1 Project descriptions

Each of the two project groups developed location-based notification systems for different clients at Virginia Tech. Notification systems are systems used in dual task situations where a user mainly focuses on a primary task while explicitly or implicitly monitoring information through a secondary system. Location-based notification systems can calculate their own location and use that information to deliver targeted information to users. Location tracking was based on the SeeVT system, which uses WiFi access point signals to determine the location of any device that has wireless access [23].

Figure 4. Screen shot of SeeVT Art showing a nearby art piece along with identifying information.

The SeeVT Art project was a new development effort whose goal was to implement a location-aware tour-guide system that would notify users of nearby art pieces and provide detailed information on request (Figure 4). The system was intended to support opportunistic navigation of the many art pieces displayed at the newly constructed hotel and conference center at Virginia Tech. The client was an employee at the Office of the University Architect.

The VTAssist project was a continuing development effort from the previous semester [3]. One of the student developers had started developing the system in the previous semester while the other was just joining the project. VTAssist is intended to help mobility impaired users locate and maintain awareness of accessible areas such as restrooms, water fountains

[1] http://www.nunit.org/

and entryways (Figure 5). The system is intended to alert users of inaccessible resources, such as when an automatic doorway stops functioning, so they can then work around those problems. Accessibility information is maintained through a collaborative feedback system. The primary client contacts were two members of the Assistive Technologies Lab at Virginia Tech.

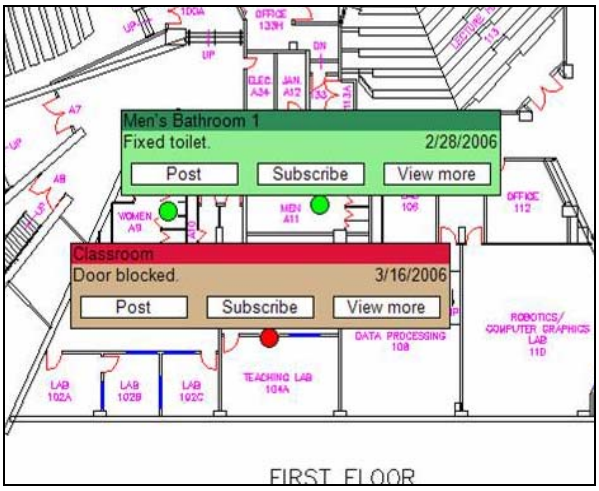

Figure 5. Screenshot of VTAssist. A top-down map view of the user's location is shown along with details about the accessibility of several locations.

4.2 Evolution of the CDR

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 allowed 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. For SeeVT Art, this included getting a better understanding of the client, learning what was to be developed and coming up with an initial story list (Figure 6). In the previous semester, the VTAssist developers developed a handheld application that would be used by wheelchair users to become aware of and navigate around accessibility problems. For this semester, they focused on developing a way to record and keep track of the accessibility problems around campus through a collaborative feedback system.

The start of each project was also where the developers set high level design goals in terms of critical parameters. These design goals for notification systems are defined by how much *interruption* the system causes, how readily it supports efficient *reaction* to the notification, and how much long-term *comprehension* the user has of information from the system [13]. These values, hereby referred to as IRC values, are typically expressed as a value between 0-1. The SeeVT Art developers determined that the IRC value for their system should be I:.5, R:.5, C: .7. This corresponds to a system with a moderate level of interruption, supports a moderate level of reaction from the user and supports a moderate to high level of comprehension. The high level of comprehension is a result of the client's desire to provide detailed information about the art pieces to foster a deeper appreciation of the art at the conference center—much of which is created by alumni. The VTAssist developers estimated the IRC of their system to be I: .35, R: .25, C: .9. The low level for interruption and reaction indicates that they and their client did not want the system to be too disruptive or intrusive to use.

1. Map Display and Location Awareness
-Develop a system to display an area map
-Display a user's location on the map

2. Opportunistic Information System
-Recognize when a user is near a POI
-Display relevant information

3. Basic Administrative Features
-Implement a mechanism through which the system may be updated

Figure 6. Excerpt from initial list of prioritized stories for SeeVT Art project.

After story development, the developers wrote out scenarios representing the key task flows identified by the clients. These scenarios were especially important for the VTAssist developers because it helped them to demonstrate to their clients how the system could rely on other Virginia Tech students and faculty to find accessibility problems which could later help mobility-impaired people. Figure 7 shows two early scenarios that demonstrate how the feedback system works.

From these scenarios, specific claims are developed that correspond to the tasks that each system needs to support and how they are supported through the interface. The claims are arranged into a *claims map*, which shows how the different tasks and interface features are related to one another. 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 (Figure 8).

John is a responsible person who liked helping other people. He is about to use the TORG elevator and observes that it has gone out of service. While deciding to use the stairs he uses **VTAssist – Tablet Edition 'feedback'** to be able to notify other users of this and save them time. After doing this he feels happy of his good deed for the day.

Tom uses a wheelchair to get around campus, he enters into TORG and uses **VTAssist – Tablet Edition 'map'** to find the closest elevator. He views the information on the elevator and discovers that he cannot use it because of the information John had left. Tom finds another elevator and is able to view directions to it through the map.

Figure 7. Two activity scenarios that describe how the system can be used to record accessibility problems and help the mobility-impaired.

Using a handheld device to navigate the art collection
+ Increases user appreciation through improved access to information
+ Users can more fully experience the art collection
+ Small form factor fits easily into user's hands
+ Intuitive touch screen interface
+ Low cost (compared to laptops)
- May interfere with user's other tasks at the center
- Small devices are easy to lose or steal
- Unfamiliar technology to average users
- handhelds have limited computing power and memory

Figure 8. Root concept claim for the SeeVT Assist project.

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 will implement a small subset of the total functionality of the system. Similarly, the CDR grows incrementally as this functionality is developed (Figure 9). The developers used IHMC CMapTools[2], a concept mapping tool, to construct the claims maps. The claim tradeoffs were not included in the diagrams in the interest of space. Observe how the CDR grows in a tree-like fashion as development proceeds.

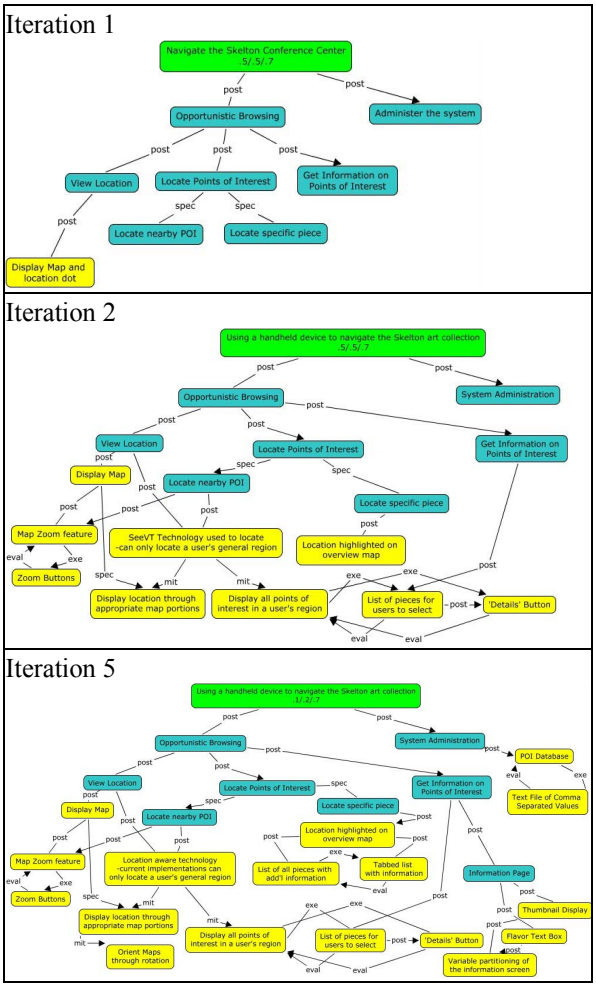

Figure 9. Claims map for the SeeVT System. The root concept claim (green) is at the top is linked to activity claims (blue) which are linked to implementation claims (yellow). Note the progressive growth through the iterations.

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

[2] http://cmap.ihmc.us/

evaluations. A detailed discussion of the relationships used in the claims maps is available here [26]. 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. For example, in the second iteration the clients and developers of the VTAssist system added the additional system task of notifying users when a previously inaccessible location becomes accessible.

4.3 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 [14]. 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.

The parallelizing of software development and usability engineering also allowed the developers to perform some processes in parallel with development. For example, the SeeVT Art project conducted a walkthrough of the conference center to gather requirements after doing a preliminary task analysis of their system and doing some initial development work. Since there were only two people in the group, these tasks were not actually conducted in parallel, but this shows how pairs could work separately and simultaneously on usability and development related tasks.

SeeVT Technology used to locate users
+ Discerns user location via WiFi signals
+ Easily used as a web service
+ Works indoors
- Does not work everywhere
- Requires wireless connection
- Not very accurate. Can only specify user's region

Figure 10. Claim describing tradeoffs of using SeeVT location tracking system

There is generally a direct mapping between a claim in the CDR claims map and any story that relates to some interface feature. There was no specific tool support that allowed the developers to manage this relationship but they were able to keep track of how they relate to each other, especially when one affected the other. Both projects used the SeeVT system to estimate locations using WiFi signals [23]. In the second iteration , the SeeVT Art project found that the SeeVT system was not accurate enough to identify individual art pieces as some were often placed very near to each other in the conference center. This required them to work with their client to develop an alternative interaction strategy to mitigate this issue. They did this by converting one of their stories into a claim that they then added to the claims map (Figure 10).

They then designed the system interaction to work around this limitation by designing the system to display a list of artwork near the user along with pictures (Figure 11). This *mitigated* the problem of accuracy in the SeeVT system and is visible in their claims map.

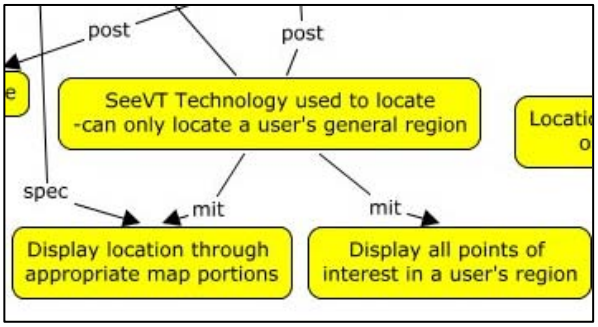

Figure 11. Portion of the SeeVT Art claims map showing how the problem of location accuracy was mitigated. (Note: only claim titles are shown above)

As shown above, technological issues can affect the way the interface is designed. The reverse also occurred in the projects. For example, during the second iteration in the VTAssist project, the clients requested that the developers implement a version of VTAssist to run on a tabletPC. This decision was largely driven by usage issues related to the handheld. The tabletPC had a larger easier to read display, would be easier to operate, and could be mounted on wheelchairs thereby freeing the users hand. This required the VTAssist developers to adjust and reorganize their ranked list of stories and refocus their development efforts.

4.4 Evaluating the interface

Evaluating the usability of the interface, unlike unit testing, is difficult to completely automate. Evaluations can collect any 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 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. For example, at the end of the third iteration, the VTAssist developers conducted a walkthrough of their system with their clients acting as proxy users. The different tasks they ran through were based on the scenarios related to the parts of the system they had focused on, while the questions and feedback they gathered were derived from the claims they wrote. The developers used colored dots (green, yellow, red), to indicate whether a location was accessible, under repair or inaccessible. During the walkthrough, the users noted that this type of indicator would be impossible for people with red-green color blindness to use. This was an unforeseen downside that they addressed in the next iteration by using indicators that relied on different shapes in addition to colors. The claim related to location status was updated and otherwise verified so subsequent evaluations did not have to focus on this area of the interface again. Lightweight usability evaluations at the end of each iteration were essentially combined with the acceptance testing process. Clients verified the functionality of the system in addition to providing feedback on its usability.

Usability evaluations can have broad impacts on the system. For example, at the end of the third iteration, the SeeVT Art developers and clients revised their overall IRC value to have very low interruption and reaction values. They determined that they wanted the system to be minimally distracting and should rely more on direct user engagement so they can focus more on the artwork itself. This led them to explore other interaction techniques such as delivering information through audio clips and providing pictures of the environment instead of an overhead map view. These quick, lightweight evaluations provided useful usability feedback without excessively limiting system development. 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. The developers were encouraged to recruit actual representative end-users to evaluate their designs—although only the SeeVT Art group was able to do this.

4.5 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.

The VTAssist team shared scenarios with their clients early in the design process to show how new task flows such as the collaborative feedback system would work. Design goals in terms of critical parameters were indirectly presented to clients. Developers would describe a system as having low interruption or distraction instead of introducing them directly to the IRC values. Claims were also not directly shown to clients. Information about specific claims was shared through the lightweight and summative usability evaluations and through general discussions during those meeting. The more detailed parts of the CDR, such as the claims map and its component claims were used by developers to iterate on their designs and weigh different design options. For example, following discussions with their client during the third iteration, the VTAssist developers wrote several claims to specify the upsides and downsides of developing on the handheld versus the tablet PC (Figure 12).

VTAssist on pocketPC	*VTAssist on tabletPC*
+ Easy to position for use.	+Larger Display
+ Lighter to move around	+Larger control area
+ Not very expensive.	+Attached to wheelchair
- Smaller display area.	- Hard to position for use
- Smaller control area.	- Heavy to carry around.
	- Significant cost

Figure 12. Claims showing tradeoffs of developing VTAssist for PocketPCs vs. tablet PCs

The CDR is the common point through which the different stakeholders in a project communicate design intentions. By linking between usability design artifacts and agile artifacts, developers are able to see the interactions between the interface design and the underlying system implementation and make appropriate tradeoffs and design decisions as necessary. The multiple perspectives of the design it shows allows for more high level discussions with clients if necessary.

5. Discussion

This section revisits the key questions about the tensions between agile processes and usability

engineering presented in Section 2.3 and details how XP+SBD addresses them (Table 1). It summarizes the conclusions from the design cases while highlighting limitations of the approach.

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 due to evaluation results or other factors. This allows 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 do 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 the claims and the 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 1. Table showing how XP+SBD approach addresses the tensions between agile methods and usability

How can developers design consistent and coherent interface architectures within an incremental agile development framework?
Incremental development of an interface supported by a design representation like the CDR + Can help developers maintain consistent and cohesive interaction design through continual evaluation of explicit design rationale and systematic improvements. + Does not limit or excessively delay incremental software delivery - Can be difficult to determine when a design representation is sufficiently complete *Using critical parameters like IRC values to guide interface development* + Can be used to measure design success through repeated evaluation of explicit metrics - Critical parameters may be difficult to define and measure
How can usability evaluations be streamlined so they better fit in accelerated development cycles while still providing useful results?
Maintaining an organized list of design tradeoffs, like a claims map, to guide lightweight usability evaluations + 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
How can project members support communication and cooperation between designers, customers, users and other stakeholders who have different backgrounds and expertise?
A shared design representation showing both high and low level views of the interaction design + Allow different stakeholder groups with different backgrounds to understand and give feedback about the design. + Can make interplay of usability and agile development work explicit and understandable + Can focus planning meetings by reminding stakeholders of key design decisions and concerns - Requires designers to actively maintain links between agile and usability artifacts

The cases also show that a design representation consisting of easy to understand artifacts that make 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.

6. Conclusions and future work

There is a need to develop ways to design software systems that can draw on the best practices and tools of different disciplines. 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 test—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.

Future efforts will build on the foundation laid out in this work and address some of the shortcomings that were identified. The researchers are currently developing a tool that allows designers to build and manage CDRs. This will help to mitigate some of the problems associated with CDR management that were experienced in this study. This tool will be built on top of an existing knowledge management framework, LINK-UP, which acts as a design knowledge repository of claims [11][20]. This will open the additional research question of how knowledge reuse through claims can support agile usability development processes.

Our next study will evaluate this approach with a practical development project through an industry partner. We will be able to evaluate how well the XP+SBD approach works in a realistic design situation. This will allow us to avoid some of the limitations of this study but will present new challenges due to the presence of additional factors will be more difficult to control.

7. Acknowledgements

We wish to thank the student developers and project clients for their valuable feedback, Miten Sampat for his help with the SeeVT location-tracking system, and Robert Biddle for his insights and encouragement.

8. References

[1] Beck, K., "Embracing change with extreme programming", *Computer*, vol. 32, no. 10, 70-77, Oct. 1999.

[2] Beyer, H., Holtzblatt, K., and Baker, L., "An Agile Customer-Centered Method: Rapid Contextual Design", *XP/Agile Universe '04*, 2004, 50-59.

[3] Bhatia, S., Dahn, C., Lee, J. C., Sampat, M., and McCrickard, D. S., "VTAssist-A location-based feedback notification system for the disabled", *in Proc. ACMSE '06*, 2006, 512-517.

[4] Carroll, J. M. and Kellogg, W. A., "Artifact as theory-nexus: Hermeneutics meets theory-based design", *in Proc. CHI '89*, 1989, 7-14.

[5] Chamberlain, S., Sharp, H., and Maiden, N., "Towards a Framework for Integrating Agile Development and User-Centred Design", *in Proc. XP '06*, 2006, 143-153.

[6] Constantine, L. L., "Process Agility and Software Usability: Toward Lightweight Usage-Centered Design." *Information Age*, vol. 8, no. 2. Reprinted in L. Constantine

(Ed.), Beyond Chaos: The Expert Edge in Managing Software Development. Addison-Wesley, Boston, MA, 2001.

[7] Cooper, A., and Reimann, R., *About Face 2.0: The Essentials of Interaction Design*, Wiley Publishing Inc., Indianapolis, IN, 2003.

[8] Greenberg, S., and Rounding, M., "The notification collage: posting information to public and personal displays", *in Proc. CHI '01*, 2001, 514-521.

[9] Hix, D., and Hartson, H. R., *Developing user Interfaces: Ensuring Usability through Product and Process*, John Wiley & Sons, Inc., New York, NY, 1993.

[10] Lee, J. C., Chewar, C. M., and McCrickard, D. S., "Image is Everything: Advancing HCI Knowledge and Interface Design Using the System Image", *in Proc. ACMSE '05*, 2005, Vol. 2, 376-381.

[11] Lee, J. C., Wahid, S., Chewar, C. M., Congleton, B., and McCrickard, D. S., "Spiraling Toward Usability: An Integrated Design Environment and Management System", Computer Science, Center for HCI, Virginia Tech, Blacksburg, VA, Tech. Rep. TR-722, 2005.

[12] Leuf, B. and Cunningham W., *The Wiki Way: Quick Collaboration on the Web*, Addison-Wesley, Boston, MA, 2001.

[13] McCrickard, D. S., Chewar, C. M., Somervell, J. P., and Ndiwalana, A., "A Model for Notification Systems Evaluation-Assessing User Goals for Multitasking Activity", *ACM TOCHI*, vol. 46, no. 3, 312-338, 2003.

[14] Miller, L., "Case Study of Customer input For a Successful Product," *in Proc. ADC '05*, 2005, 225-234.

[15] Nelson, E., "Extreme Programming vs. Interaction Design", *Fawcette Technical Publications*, 2002. http://www.fawcette.com/interviews/beck_cooper/.

[16] Newman, W., "Better or Just Different? On the Benefits of Designing Interactive Systems in terms of Critical Parameters", *in Proc. DIS '97*, 1997, 239-245.

[17] Norman, D. A., "Do companies fail because their technology is unusable?", *Interactions*, vol. 12, no. 4, 69, 2005.

[18] Olsen, G., "The emperor has no lab coat", *Interactions*, vol. 9, no. 4, 13-17, 2005.

[19] Patton, J. "Hitting the target: adding interaction design to agile software development", *in Proc. OOPSLA '02*, 2002, 1-ff.

[20] Payne, C., Allgood, C. F., Chewar, C. M., Holbrook, C., and McCrickard, D. S., "Generalizing Interface Design Knowledge: Lessons Learned from Developing a Claims Library", *in Proc. IRI '03*, 2003, 362-369.

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

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

[23] Sampat, M., Kumar, A., Prakash, A., and McCrickard, D. S., "Increasing Understanding of a New Environment using Location-Based Notification Systems", poster paper *in Proc. HCII '05*, 2005.

[24] Sharp, H., Biddle, R., Gray, P., Miller, L., and Patton, J., "Agile development: opportunity or fad?", *In proc. extended abstracts CHI '06*, 2006, 32-35.

[25] Sutcliffe, A. G., "On the Effective Use and Reuse of HCI Knowledge", *ACM TOCHI*, vol.7, no. 2, 197-221, 2000.

[26] Wahid, S., Allgood, C. F., Chewar, C. M., and McCrickard, D. S., "Entering the Heart of Design: Relationships for Tracing Claim Evolution", *in Proc. SEKE '04*, 2004, 167-172.

[27] Wahid, S., and McCrickard, D. S., "Claims Maps: Treasure Maps for Scenario-Based Design.", *In Proc. ED-MEDIA '06*, 2006, 553-560.