Distributed and Automated Usability Testing of Low-Fidelity Prototypes

Ali Hosseini-Khayat, Theodore D. Hellmann, Frank Maurer
Department of Computer Science
University of Calgary
Calgary, Canada
{hosseisa, tdhellma, maurer} @ cpsc.ucalgary.ca

Abstract – Lack of tool support is hindering the growing interest in incorporating usability evaluation into agile software development practices. To address this concern we have developed ActiveStory Enhanced, a tool for creation and remote evaluation of low-fidelity prototypes. Building on its predecessor, ActiveStory Enhanced remotely collects more forms of usability data and provides new visualizations of this data to aid in the detection of usability flaws.

Keywords – prototyping; usability testing; agile

I. INTRODUCTION

Tool support for automated, distributed usability testing of low-fidelity (low-fi) prototypes has the potential to help mitigate some of the problems posed when conducting tests in an agile context, but is currently quite limited. Furthermore, data collected from usability testing plays an important role in detecting usability issues in a user interface, however support for capturing and presenting this data in a meaningful way is weak in existing tools. Wizard of Oz (WOz) testing is a type of usability testing that is performed on prototypes of a system. This form of testing is of particular interest in the context of usability testing in agile teams, due to the short timeframes between releases, difficulty and cost of gathering participants, and cost of execution. We attempt to address the limitations of existing prototyping and Wizard of Oz-like testing tools, particularly in the context of agile usability engineering, by developing a toolset targeted at that usage.

A. Usability

The ease of interacting with a software system is referred to as the usability of the system. The usability of an interface can be a determining factor in the success or failure of software. Software that is not built with usability in mind can lead to rejection of the software by end-users or reduction in the efficiency of users’ workflow. Usable software is more likely to be successful in terms of acceptance by end-users. It can also lead to improved efficiency and even encourage users to learn more advanced features or functionality of a system that may otherwise have been missed or ignored. Many methods and processes have been developed to help improve the usability of an interface including prototyping [1], [2], contextual inquiry [3], heuristic evaluation [4], [2] and usability testing techniques [2] like Wizard of Oz testing [5].

B. Agile Methods, Conflicts with Usability Engineering

Given that usability is a component of high-quality software, there is significant interest in applying it in an agile context. One of the core concepts of agile methods is responsiveness to changing requirements, rather than obtaining and adhering to design documents. This mindset clashes with the inherent need of usability engineering techniques for large amounts of design and evaluation of prototypes prior to implementation. The discrepancy between the two comes from the preference towards upfront design in usability engineering, versus minimal upfront design in agile methods. The lack of attention to usability has become a growing concern in the agile and usability communities (as evidenced by increasing traffic on the agile-usability Yahoo group [6]).

Attempts have been made to make usability engineering a more agile practice [7], [8], [9], [10], [11], based on previous work on discount usability by Jakob Nielsen [12]. Discount usability evaluation was designed with the idea that some evaluation is better than none and that scientific evaluation is too costly in most cases. Further research has shown that the most problematic usability issues can already be identified when dealing with a wireframe, sketch-like prototype (known as a low-fi prototype) [13]. These types of prototypes help the designer or a test participant focus on more critical aspects of the prototype such as the logic, flow and placement of interface elements, rather than superficial details such as color and font.

Wizard of Oz (WOz) testing is a form of usability testing designed to help test low-fi prototypes on sample users [5] by having them interact with the sketched prototypes, in the same way that they would interact with a fully functional interface. As part of WOz testing, a (human) designer known as the
Wizard simulates the role of the computer system. The test participant is shown a page of a prototype and asked to interact with it in order to achieve a given task. Once the test participant chooses the interaction they want to do, the Wizard transitions to the page of the prototype which shows the result of the interaction. Typically, a third person observes the users interactions and notes any difficulties in terms of flow, continuity, logic and comprehension [5].

WOz testing can still be costly and time consuming due to the amount of preparation that must be done prior to execution and evaluation – especially gathering participants, bringing them on-site and performing the test [14]. This is an issue for agile teams due to the short time between iterations and releases, making it difficult to perform these time consuming usability tests as frequently as needed. Performing these tests in a distributed fashion without collocating the test participants and conductors could reduce the overhead of organizing participants and the financial costs involved in performing usability tests. Since this style of testing differs from WOz testing, in that there are no humans involved in conducting the test, we refer to it as low-fi testing (LoFT).

C. Tool Support for Agile Usability

Currently, tool support for usability evaluation is limited in terms of support for LoFT. While many tools are available that support creation of low-fi prototypes, only some of these can produce automated prototypes and even fewer allow distributed testing. Tools that aid in conducting Wizard of Oz testing also exist, however they do not provide prototyping support or automated transitioning.

With the exception of the original ActiveStory by Patrick Wilson [15], none of these tools are catered to an agile usability process which specifically includes running them in a distributed fashion and collecting data. Due to its focus on supporting executable low-fi prototypes, the original ActiveStory is limited to basic data collection.

II. RELATED WORK

Multiple published papers report on agile teams that have attempted to integrate user interaction design in their process in practice in an attempt to produce more usable applications. These cases consist of work by Larry Constantine [7], Jeff Patton [8], Gerard Meszaros [11], Desiree Sy [9], and Jakob Nielsen [10]. In some cases, the authors were initially usability professionals who then applied agile methodologies to their design process (such as Constantine and Sy). In other cases the authors were agile practitioners who integrated usability engineering into their practices. In both cases the resulting processes share many commonalities.

A. Remote Usability Evaluation

Usability testing has also been performed by having users remotely logging on to a computer at the developer’s site and performing tasks on an application under development. This application can be instrumented in order to record usability data. Hartson et al. [16] found that when users were allowed to initiate this recording themselves, trivial usability issues were filtered out, allowing effort to be focused on serious flaws.

Similarly, usability testing can be performed on websites by recording the path a user takes to perform a task. WebQuilt [17] redirects requests to a proxy server, which is able to log this information before relaying these requests to the actual webserver. A diagram of the user’s path is provided by the tool for usability analysis.

B. Prototyping

Constantine first presented the concept of Usage-Centered Design in a paper of the same name [18], [7]. This process involves evaluating a low-fi prototype and refining it. The paper also suggests that while a Big Design Up Front approach is not necessary, some minimal up-front design for user interfaces is still a necessity in order provide a consistent and well-organized interface for users. Patton describes a practical application of Constantine’s process in [8]. A concept similar to WOz testing is introduced in which members of the team play both the roles of the system and users after creating the prototypes in order to provide validation. Sy [9] presents her usability team’s experience with adopting agile methods. As part of their process, prototypes were iteratively developed and refined. At the end of each iteration, the prototype would be evaluated by test users with increasing similarity to end users. A 2006 experience report [11] by Meszaros describes an increase in end user buy-in and acceptance after the addition of some prototyping and usability evaluation to an agile project. Sharp et al. [19] also conducted a field study to investigate the integration of agile methods and UCD in an organization. One of the five principles they recommend for integration is prototyping.

C. Existing Tool Support for Low-Fidelity Testing

Some tools used for interaction design were specifically developed for that purpose (like Microsoft Expression Blend), while others are simply general-purpose tools (like Microsoft Visio and Microsoft PowerPoint). Currently, the only prototyping tool designed specifically with agile interaction design in mind is the original ActiveStory.

The most commonly-used tools for prototyping are physical pens and paper. Obvious advantages of pen and paper are that they’re cheap, natural and easy to use, and a prototype can be sketched in a matter of
minutes. Moreover, making changes is as easy as erasing the portion that needs to be changed and redrawing it. Also, it’s easier for developers to make changes to prototypes, even in the middle of a test session, because they are quickly created with minimal effort and emotional attachment. However, they are difficult to share with others, since they are not in digital form. It can also be difficult to view the paper prototype as a future computer system due to the less than convincing look of sketches.

SketchFlow [20] is a low- to medium-fi prototyping environment. Prototypes can be created using a series of UI elements designed to look like hand-drawn sketches. A flow can also be specified for a prototype, allowing transition between different states through interactions such as button clicks. Designers can also specify interactions such as dragging and dropping, animations and selection. The prototype can be interacted with and annotated by test users. Prototypes can easily be transitioned into a higher level of fidelity and even switched over to a production design. In this way time can be saved by reusing an approved prototype design. At the moment, this is the only tool that provides potential for reuse. SketchFlow, however, does not allow distributed testing and does not collect any usability data other than comments and annotations.

The original ActiveStory is a tool targeted specifically towards agile interaction designers [15]. The tool is designed to allow sketching prototypes as on paper. A flipchart metaphor is used for organizing pages of the prototype. Pages of the prototype can be linked to other pages by adding interaction hotspots to the drawings. The prototype can then be exported to a built-in web server which hosts the WOz testing portion of the tool. While the data collected by the WOz portion is useful, it was left in its most basic form without any analysis or visualizations to help support it.

WoZPro [21] is a tool that allows freehand drawing on a canvas, similar to ActiveStory. It introduces the ability to create master pages, which automatically update any pages that are based on it, if it is edited. WoZPro prototypes execute within the tool in a slideshow manner, with no interaction and no usability data collection.

SILK [22], [23], DENIM [24], Serena Prototype Composer [25], Microsoft Visio [26] and Balsamiq Mockups [27] all support creation of prototypes by dragging and dropping widgets onto a canvas. These prototypes can be interacted with by test users but don’t collect any usability data and don’t allow distributed testing. Serena Prototype Composer is more applicable to high-fi prototyping while the others are targeted at low-fi.

Microsoft PowerPoint [28] is sometimes used for prototyping purposes. It is possible to package slides and distribute them, but they can’t be executed remotely and no usability data is collected.

Morae is a commercial tool for usability testing developed by TechSmith [29]. Morae uses commodity hardware for the purpose of recording audio, video and screen captures. It provides an entire suite of usability test administration tools including surveys, calculation of usability metrics and recording audio and video. However, Morae tests need to be conducted on a working prototype or interface, or another tool must be used to perform the WOz functionality.

III. TOOL REQUIREMENTS

In this section requirements for the tool are outlined, as well as how the requirements were obtained.

A. Gathering Requirements

Requirements were gathered based on a survey conducted previously by Patrick Wilson [15] for the original ActiveStory. Web survey participants were recruited from the agile-usability Yahoo! Group [6], which has a significant membership of agile and usability practitioners. A number of them were also selected for interview-based survey, which was aimed at establishing what kind of tool agile usability practitioners would find useful and practical. The requirements were extracted with an integrated agile and usability team in mind. As such, they served as an appropriate basis for the requirements of the new tool, ActiveStory Enhanced (ASE). ASE must:

- Be easy to use
- Preserve the pen and paper metaphor
- Quickly create and test prototypes
- Allow design flexibility
- Provide prototype interactivity
- Allow remote, distributed testing
- Collect data from distributed tests

Supplementary requirements for the tool were also obtained from analyzing the results of the evaluation done on the original ActiveStory tool. Firstly, ASE must accommodate the differences between desktop and tablet PCs, namely the angle of the screen (vertical vs. nearly horizontal) and the partial obstruction of the screen on tablet PCs. The next requirement was to provide the user with the conveniences of a digital drawing experience, such as copy, paste, undo, redo and page duplication. The user should also have greater control over the pen tip and eraser, in terms of shape, size and even angle. The
final requirement was to provide a simpler metaphor for the prototype, with a view of the big picture. The tool should have support for visualizing and analyzing collected usability data in a meaningful way.

IV. **ACTIVE STORY ENHANCED**

ASE is an open source toolset designed to help agile teams create prototypes, perform LoFT and review usability data in an affordable and quick way in line with agile practices. It is available for download via CodePlex (Link: [30]). Improvements to the original ActiveStory include but are not limited to: live mouse trails; a wider range of visualizations; and usability data and support for common digital drawing functionality such as copying, pasting, undoing and duplicating pages. ASE allows freehand sketching of user interface prototypes in the Prototype Designer. It also allows usability evaluation of prototypes by remote users through the Testing Tool. The Test Reports tool provides different forms of usability data and visualizations as well as some analysis functionality.

A. **Prototype Designer Overview**

An ASE prototype from the point of view of the Prototype Designer tool is essentially a series of (usually black and white) unordered images made up of strokes sketched by the designer, images imported from external files and interaction links (or hotspots) added by the designer. This system is simple to learn while at the same time providing creative freedom, as evidenced by the study performed by Patrick Wilson for the original ActiveStory [15].

The metaphor in ASE is a set of loose pages which can be quickly viewed and modified, similar to how a designer might have a number of physical papers loosely arranged on their desk which show the different visual states of the system.

![Figure 1. ASE Prototype Designer](image)

The optimal ActiveStory Enhanced experience with the Prototype Designer is on a Tablet PC. As the tool was developed to natively utilize the Tablet PC SDK, rather than relying on mouse data, the design process, particularly drawing, is very smooth and natural. Pressure sensitivity on Tablet PCs allows for more precise drawing. The stylus can also be inverted and used as an eraser. Together these features contribute to a more natural drawing experience. Additionally, the UI is laid out horizontally in order to optimize the canvas real estate. Other drawing devices can also be used to draw ActiveStory Enhanced prototypes, including mice, drawing tablets, digital whiteboards and digital surfaces. Images in various formats can also be directly imported.

Operations that are typically not possible with pen and paper are also available to the user with the conveniences of a digital drawing experience. This includes the ability to undo and redo strokes as well as copying, pasting, deleting, lassoing, resizing and moving elements. The Prototype Designer also allows creating carbon copies of entire pages. This could be useful when drawing different states of a base page, for example the different drop-down menus of an application prototype.

Interactions are added to prototype pages by specifying regions of the page, which redirect to another page when clicked by test participants. In this way a prototype can be fully automated, removing the need for a human Wizard.

Once the prototype has been designed it is exported to the Testing Tool which generates and administers a LoFT test. The prototype can be exported to a built-in web server that is packaged as part of ASE, the Microsoft IIS Web Server, an existing ASP.NET server or any existing Java Servlet server.

B. **Testing Tool Overview**

The Testing Tool enables usability experts to perform LoFT tests on prototypes that are designed and exported from the Prototype Designer. The Testing Tool allows participants to remotely perform usability testing via the Web, removing the need for travel and reducing the time and financial costs of taking part. Test execution costs are also reduced – for example a “wizard” is no longer necessary and an observer is optional. Gathering participants for usability tests is significantly easier as the time commitment made by participants is reduced. It’s also possible to recruit a large body of participants from around the world due to the wider potential audience. In order to administer the usability test, the usability expert need only distribute the URL of the LoFT test.

Participants are greeted with a welcome dialog that instructs them on using the LoFT test. Task instructions set forth by the usability expert are also presented to the participant at this point. Users can
access the task instructions at any point in time during the LoFT test.

The usability test begins when the participant clicks the “Start” button, from which point onward time will be tracked for the purpose of calculating time spent on pages. The user can temporarily pause the session at any time, which avoids artificially high elapsed time data. Once the test session has started the user is shown the first page of the prototype (as specified by the tester when exporting) and can begin to interact with it. Throughout the entire session, usability data is collected automatically. The participant has the option to leave feedback in the form of comments or emoticons at any point during the session.

C. Test Reports Tool Overview

The final tool in the ASE toolset, the Test Reports tool, allows reviewing the collected usability data. The Test Reports web application provides multiple views and visualizations of the collected data as well as some basic computer-assisted analysis.

Mouse trails are a representation of where a participant’s mouse has moved and where they’ve clicked. Chen et al. [31] report that in most cases mouse movements correspond with the location on the screen which the user is looking at and considering. Therefore, a user’s eye movement can be approximated in a lightweight and cheap way with acceptable accuracy [31] by tracking mouse movements. Furthermore, the amount of data collected is small enough to be able to transmit over the web. One of the drawbacks of this technique is that the correlation between the eye- and mouse-movements is dependent on the type of application. For example, a word processor will have a weaker correlation than an application that relies more on the mouse, such as web browsing. Also, this correlation and the accuracy of this technique vary between different people.

The Reports tool allows reviewers to see each page visited during any given test session augmented with mouse trails and clicks. As well, a live representation is provided which traverses the pages visited and reenacts the mouse movements and clicks with the original speed. This essentially is essentially the same information available via a recording of the participant’s screen, without actually using screen capture and incurring heavy bandwidth usage.

A view of each page with the aggregated trails and clicks for all test sessions is also available. Trends in participants’ movements and clicks could potentially be identified in this way. Each page is also accompanied by details that may aid in finding trouble spots including the number of users who visited, comments left by users, number of clicks, average number of clicks per page visit and per user, total amount of time spent across all users, average duration spent per visit and per user. The tool also visualizes the captured statistics across all test instances in the form of charts. These charts can be helpful in identifying points of interest for further investigation at a glance, rather than manually going through all of the statistics and finding information that stands out. For example if the average amount of time spent on a page per page visit is much higher than most other pages, it may be that a usability issue exists on that page (although this is not necessarily always the case).

The analysis section provides the reviewer with the option to perform either “Expected Path Analysis” or “Comments Search” on the collected data. Expected Path Analysis allows the reviewer to specify a given path through the prototype and search for any test instances which do or do not fully or partially match. This can be useful in identifying users who deviate from the expected path for further review. Comments search works in a similar way.
V. TOOL ANALYSIS

In order to determine how useful the tool is for agile usability evaluation, a comparison with existing tools was performed. Criteria for this were selected based on the needs of agile usability evaluation. The first criterion is the ability to execute usability tests, which is met by any tool that allows viewing and interacting with prototypes. The second criterion is the ability to perform remote tests, meaning the prototype should be viewable and allow interactions in a distributed environment such as the Internet. The ability to collect usability data is the third criterion, and the fourth is the ability to view, analyze, and visualize the data. Table I presents the result of the comparison.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Executable Prototypes</th>
<th>Remote Usability Testing</th>
<th>Automated Usability Data Collection</th>
<th>Usability Data Analysis / Visualization</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveStory Enhanced</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Paper and Pen</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SketchFlow</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ActiveStory Original</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SILK</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Balsamiq</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PowerDirector</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wireframe</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AxureRP [32]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Morae</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

In this table, a check mark (✓) indicates that the criteria is met completely, a cross mark (✗) means the criteria is not met and a fraction mark (½) indicates that the criteria is only partially met.

As the table indicates, only ASE, ActiveStory Original, and SketchFlow meet all of the criteria. Another comparison was done between tools that support remote usability data collection in terms of the types of data they collect. The following table presents a comparison between ASE, original ActiveStory [15], Morae [29] and SketchFlow [20].

<table>
<thead>
<tr>
<th>Data Type</th>
<th>ActiveStory Enhanced</th>
<th>ActiveStory Original</th>
<th>SketchFlow</th>
<th>Morae</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participant Comments</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Audio Feed</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Mouse Trails and Clicks</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Live Session Playback</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Video Feed</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aggregated Mouse Trails</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Expected Path Analysis</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Text Visualizations</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Heuristic Evaluation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Eye Tracking</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Statistic Charts</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

As the table shows, ASE provides a more comprehensive set of collected usability data than the original ActiveStory or SketchFlow. Although Morae provides support for video and audio feeds as well as performing heuristic analysis, it does not do so over the Web and is limited to a local network. Additionally, these functionalities are not an ideal fit for a lightweight approach to usability testing. We think the data provided by ASE, as well as its visualizations, should help increase the number of usability issues detected as well as easing their detection. An evaluation of these visualizations was conducted, and will be published at a later time. Case studies of the applicability of the tool were not conducted, as this had already been done for the original ActiveStory.

VI. STRENGTHS, WEAKNESSES, AND LIMITATIONS OF ACTIVESTORY ENHANCED

One of the main strengths of ASE is its support for remotely collecting usability data, which allows for greater flexibility when recruiting participants without sacrificing time, financial resources, or useful feedback. This is also good for conducting usability tests on a global level, for instance for the purpose of localization.

The creative freedom and flexibility that the Prototype Designer tool provides is also a major strength of ASE, as it gives designers greater freedom of expression without imposing rules and UI design concepts on them. It also makes it easier to develop a prototype design by eliminating the need to learn the interface and available widgets. In fact, in ASE, a
widget can look like anything the designer wants, which is especially important in the design of non-standard user interfaces or new interactions, such as touch interactions or interfaces for mobile devices.

Arguably one of the weaknesses of ASE is its lack of support for widget recognition or drag-and-drop interface design (GUI builder functionality). This is useful when designing a standard interface since most of the widgets needed would be readily available. This system also has the advantage of leveraging widgets with pre-defined and well-known behavior, a characteristic which could be used to expand interaction support.

Lack of visualization for comments is one of the limitations of the Reports tool. Examples include frequency-based visualizations such as those commonly used for tags on the web. Lack of support for collecting audio and video data is another limitation of ASE. However, the need for this data in an agile context is questionable due the time required for processing this information. Nonetheless, the utility of video and audio data is undeniable and for collecting audio and video data is another limitation of the Reports tool. Examples include frequency-based visualizations such as those commonly used for tags on the web. Lack of support for collecting audio and video data is another limitation of ASE. However, the need for this data in an agile context is questionable due the time required for processing this information. Nonetheless, the utility of video and audio data is undeniable and ideally this data would be remotely collected. Finally a major limitation of ASE is that it does not provide any way of testing touch-based interactions, such as scaling or rotating widgets, or other interactions, such as scrolling. Given the recent attention to touch-based devices it would be useful to support touch-based interactions in prototypes.

VII. FUTURE WORK

Further work on ASE involves extending it to support touch-based interactions, as well as transitioning between different levels of fidelity. An evaluation can also be done to compare the efficacy of audio and video data versus the data collected by ASE in detecting usability flaws. Research is also in progress which involves test-driven development of user interfaces, in which ASE plays an important role.

VIII. REFERENCES

[23] James A. Landay and Brad A. Myers, "Sketching


