Combining agile methods and user-centered design to create a unique user experience: An empirical inquiry

Cynthia Y. Lester
Department of Computer Science
Tuskegee University
Tuskegee, Alabama, USA
cylester@tuskegee.edu

***Abstract* - With the advent of the Internet and websites, many people believe that website development is as easy as dragging an icon here, placing a menu there, and adding a picture. However, there is more to website design than many people believe especially if you desire to develop a website that meets the needs of the user and follows software engineering principles. While there are many software process models and human-computer interaction activities that focus on the user, the integration of these activities is quite difficult, especially as it relates to website development. This paper presents the results of an empirical investigation that combined one activity of human-computer interaction, user-centered design, and one software engineering method, agile development into a small-scale development exercise that specifically focused on website development. The results from the study suggest that using the hybrid approach for small-scale projects is easy to implement, but is not without challenges.**

Keywords – agile development; human-computer interaction; software engineering; user-centered design

I. INTRODUCTION

The use of technology and the Internet is commonplace in today's society. In 1990, it was reported that there were less that 50 million users of the Internet in the U.S. However, by 2008 the U.S. reported approximately 230,630,000 Internet users [1]. Therefore, it stands to reason that with more users and more advanced systems, the user population of today's technology would be more technically savvy than those user groups of yesteryear. However, the average user is now less likely to understand the systems of today as compared to the users of a decade ago. Consequently, the designers and developers of these systems must ensure that the systems are designed with the three "use" words in mind so that the system is successful. Hence, the system must be useful, usable, and used [2]. The last of the "use" terms has not been a major factor until recently, thereby making the discipline of human-computer interaction increasingly more important.

Human-computer interaction (HCI) has been described in various ways. Some definitions suggest that it is concerned with how people use computers so that they can meet users' needs, while other researchers define HCI as a field that is concerned with researching and designing computer-based systems for people [3], [4]. Still other researchers define HCI as a discipline that involves the design, implementation and evaluation of interactive computing systems for human use and with the study of major phenomena surrounding them [5]. However, no matter what definition is chosen to define HCI, the concept that all these definitions have in common is the idea of the technological system interacting with users in a seamless manner to meet users' needs. Consequently, system developers need to further their understanding of the human, the user, and the interaction.

The aim of this paper is to present the results from an empirical inquiry that combined one activity of HCI, user-centered design, and one software engineering method, agile development, to develop a website for a small-sized business. The paper also touches on the theme of extreme programming as the implementation methodology for agile methods. While there are many different development strategies specifically for website design and development, a review by the author revealed that there was little consistency among the processes and some did not address user involvement or the user experience. Therefore, a hybrid approach using agile development and user-centered design was considered since both focus on the inclusion of the user throughout the development process.

The paper is divided into the following sections: the human, the system, and the interaction; traditional software methodologies; agile methods; user-centered design; a practical implementation combining the two methods; a discussion of the empirical investigation; and concluding thoughts. It is the desire of the author that the readers of the paper will see how closely related the two methodologies are and how they can be used together for small software development projects that yield high levels of user involvement while creating an enriched user and developer experience.

 ISBN: 978-1-61208-117-5

II. THE HUMAN, THE SYSTEM, AND INTERACTION

A. The human user

The human user may be an individual or a group of users who employ the computer to accomplish a task. The human user may be a novice, intermediate, or expert who uses the technological system. Further, the human user may be a child using the system to complete a homework assignment or an adult performing a task at work. Additionally, the human user may be a person who has a physical or cognitive limitation which impacts his/her use with the computer-based system. No matter who the human user is, the goal when interacting with a computer system is to have a seamless interaction which accomplishes the task.

B. The computer

According to the *Random House Unabridged Dictionary*, a computer is defined as an electronic device designed to accept data, perform prescribed mathematical and logical operations at high speed, and display the results of these operations [6]. However, as computers become more complex, users expect more than just a display of the results of their operations. The term computer system is used to represent technology and technological systems. Consequently, technology or technological systems encompass many different aspects of computing. Users now require their systems to be able to provide answers to questions, to store various forms of information such as music, pictures, and videos, to create a virtual experience that physically may be unattainable, and to understand verbal, visual, audio, and tactile feedback, all with the click of a button. As the human user becomes to depend on these technological systems more, the interaction between the user and the system becomes more complex.

C. The interaction

Interaction is the communication between the user and the computer system. For computer systems to continue their wide spread popularity and to be used effectively, the computer system must be well designed. According to Sharp, Rogers, and Preece, a central concern of interaction design is to develop an interactive system that is usable [4]. More specifically, the computer system must be easy to use, easy to learn, thereby creating a user experience that is pleasing to the user. Consequently, when exploring the definition of interaction, four major components are present which include:

- The end user
- The person who has to perform a particular task
- The context in which the interaction takes place
- The technological systems that is being used

Each of these components has its own qualities and should be considered in the interaction between the computer system and the user. In his bestselling book, *The Design of Everyday Things*, Donald Norman writes about these components and how each must interact with the other, suggesting that the common design principles of visibility and affordance help to improve interaction [7]. The principle of visibility emphasizes the idea that the features of the system in which the user interacts should be clearly visible and accessible to human sense organs, which improves the interaction between the action and the actual operation [7]. The principle of affordance as suggested by Jef Raskin, should accommodate visibility such that the method of interacting with the system should be apparent, just by looking at it [8].

Therefore, in order to create an effective user experience, a designer of an interactive computer system must understand the user for which the system is being created, the technological system that is being developed and the interaction that will take place between the user and the computer system. However, traditional plan-driven software engineering methodologies often make integrating the user into the development process to achieve an effective user experience difficult.

III. TRADITIONAL SOFTWARE METHODOLOGIES

Software engineering is defined as "being concerned with all aspects of the development and evolution of complex systems where software plays a major role. It is therefore concerned with hardware development, policy and process design and system deployment as well as software engineering [9]."

The term software engineering was first proposed at the 1968 NATO Software Engineering Conference held in Garmisch, Germany. The conference discussed the impending software crisis that was a result of the introduction of new computer hardware based on integrated circuits [9]. It was noted that with the introduction of this new hardware, computer systems were becoming more complex which dictated the need for more complex software systems. However, there was no formalized process to build these systems which put the computer industry at jeopardy because systems were often unreliable, difficult to maintain, costly, and inefficient [9]. Consequently, software engineering surfaced to combat the looming software crisis.

Since its inception, there have been many methodologies that have emerged that lead to the production of a software product. The most fundamental activities that are common among all software processes include [9]:

- *Software specification* – the functionality of the system and constraints imposed on system operations are identified and detailed
- *Software design and implementation* – the software is produced according to the specifications

 ISBN: 978-1-61208-117-5

- *Software validation* – the software is checked to ensure that it meets its specifications and provides the level of functionality as required by the user
- *Software evolution* – the software changes to meet the changing needs of the customer

The activities that formulate this view of software engineering came from a community that was responsible for developing large software systems that had a long life span. Moreover, the teams that used this methodology were typically large teams with members sometimes geographically separated and working on software projects for long periods of time [9]. Therefore, software development methodologies that resulted from this view of software engineering were often termed as "heavyweight" processes because they were plan-driven and involved overhead that dominated the software process [9]. However, great difficulty occurs when these methodologies are applied to smaller-sized businesses and their systems, because these methods lack the agility needed to meet the changing needs of the user. The next section presents an overview of an alternative to heavyweight processes, agile development.

IV. AGILE METHODS

In an effort to address the dissatisfaction that the heavyweight approaches to software engineering brought to small and medium-sized businesses and their system development, in the 1990s a new approach was introduced termed, "agile methods." Agile processes are stated to be a family of software development methodologies in which software is produced in short releases and iterations, allowing for greater change to occur during the design [10]. A typical iteration or sprint is anywhere from two to four weeks, but can vary. The agile methods allow for software development teams to focus on the software rather than the design and documentation [9]. The following list is stated to depict agile methods [9], [10]:

- *Short releases and iterations* - allow the work to be divided, thereby releasing the software to the customer as soon as possible and as often as possible
- *Incremental design* – the design is not completed initially, but is improved upon when more knowledge is acquired throughout the process
- *User involvement* – there is a high level of involvement with the user who provides continuous feedback
- *Minimal documentation* – source code is well documented and well-structured
- *Informal communication* – communication is maintained but not through formal documents
- *Change* – presume that the system will evolve and find a way to work with changing requirements and environments

More specifically, the agile manifesto states:

"We are uncovering better ways of developing software by doing it and helping others to do it.
Through this work we have come to value:
Individuals and interaction over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan
That is, while there is value in the items on the right, we value the items on the left more."

While agile methods are considered as lightweight processes as compared to their predecessors, it has been stated that it sometimes difficult especially after software delivery to keep the customer involved in the process [9]. Moreover, for extremely small software projects, the customer and the user may be one in the same, further complicating the development process. Therefore it is of interest to consider HCI, particularly user-centered design and the benefits it may have if combined with agile methods for software development. The next section introduces the concept of user-centered design.

V. THE USER-CENTERED DESIGN PROCESS

A central theme in HCI is to make the focus of design activity, 'user-centered'. According to human centered design processes for interactive systems, ISO 13407, "Human-centered design is an approach to interactive system development that focuses specifically on making systems usable. It is a "multi-disciplinary activity" [11]. User-centered design (UCD) tends to lead to fewer errors during development and lower maintenance costs over the lifetime of the computer software [12].

In contrast to the traditional methods of software development, user-centered design aims at understanding the user and designing the user interaction through an iterative process. At the center of user-centered design is the user with requirements emerging from user interaction with the system. Since the user-centered design process is an interactive one which allows users to interact with system designers to design a system, ultimately the needs of the user are met.

There are four basic components which help to define interaction [13]. Those components include:

- The end user
- The person who has to perform a particular task
- The context in which the interaction takes place
- The technological systems that is being used

Each of these components has its own qualities and should be considered in the design of the system. The UCD process allows for the exploration of each of these components. As with most methodologies, the UCD

 ISBN: 978-1-61208-117-5

process can be broken down into four steps. These steps are analysis, design, implementation and deployment, and are shown in figure 1.

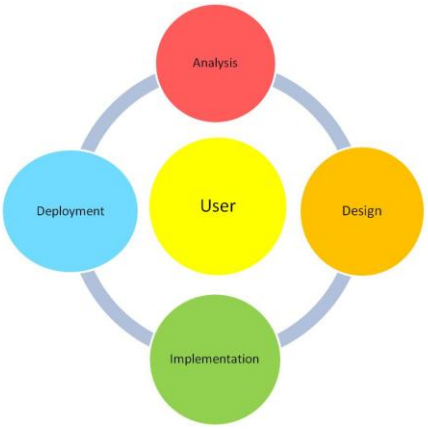

Figure 1. User-centered design model

VI. A PRACTICAL IMPLEMENTATION OF THE AGILE METHOD AND UCD

A. *The project*

The purpose of the project was to combine the principles found in user-centered design with the agile manifesto to develop a website for a customer who was also part of the user group.

B. *The stakeholders*

The stakeholders consisted of two groups: the customer who commissioned the project and the user group who consisted of selected parents and students. The customer was a program manager for a grant obtained to fund a Research Experience for Undergraduates (REU) in an integrative biosciences program at a mid-sized university. The customer has some technical expertise and expressed a desire to be involved in the entire development process. Therefore, to ensure that the customer who was also part of the user group was at the center of the process, bi-weekly meetings were established where updates were provided and prototypes were presented.

C. *The development team*

The development team consisted of two programmers who have expertise in website development and the principles of UCD.

D. *The implementation*

Extreme programming (XP) is probably one of the best known and most widely used agile methods [14], [15]. It was originally designed to address the needs of software development by small teams who faced changing requirements and system environments. XP was used in this empirical inquiry because it reflected the four following principles:

- Incremental development is supported through small, frequent releases
- Customer involvement is integral and supported throughout the process
- People are the main focus of the process not the development process
- Change is embraced as prototypes were constantly released to the user
- The design for the website was simple

XP was also used because it incorporates the concept of collaborative working. The most extensively investigated practice of XP is perhaps pair programming.

The basic premise of pair programming is that a pair of developers, work together during the development process. The developers sit as the same computer and develop the software. There have been several studies that have confirmed that pair programming is effective and can lead to better quality software [9]. However, some studies suggest that with more experienced programmers there is a loss of productivity [16]. Further in a study of nearly 500 students it was found that the stronger of the pair did most of the work, while the weaker of the pair did not improve in programming skill [17].

Yet, it was decided that pair programming would be used because it fosters communication between the team members working on the website and it supports the idea of collective ownership and responsibility. Moreover because the team consisted of only two members with similar programming backgrounds, pair programming proved to be a natural fit.

The first step in the project was to design user stories. User stories are requirements which can be implemented into a series of tasks [9]. User stories are often thought of as high-level requirement artifacts. There are several things to consider when developing user stories which include [18]:

- Stakeholder/customers write the user stories
- Simple tools like index cards to capture thoughts should be used
- The stories can be used to describe a variety of requirements
- Time for the pair programmers to implement the story should be considered
- Priority regarding implementation should be considered

In order to develop the user stories, the team met with the user group who supplied the information and the content

 ISBN: 978-1-61208-117-5

for the website. An example of a user story that was created for the website is found in figure 2.

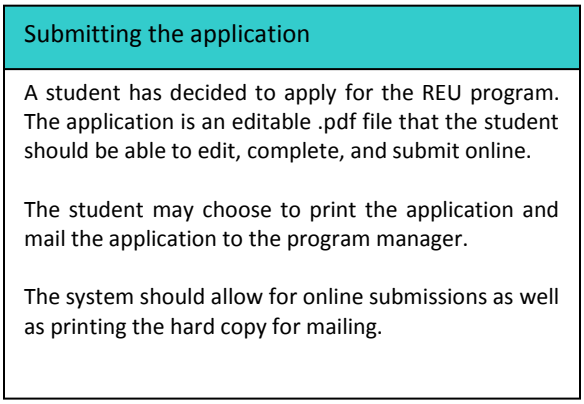

Figure 2. Story card

After the user stories were developed, the story cards were broken into tasks and the user group was asked to organize the tasks according to priority of what should be implemented first. The objective of this step was to determine the resources needed for implementation. At the completion of this planning process, there were approximately twenty story cards with varying requirements which were organized according to priority.

In the next phase of the project, the development team began implementing the stories according to priority. It was imperative to the customer that the application for the program be the first item implemented. Once this was implemented, the prototype was delivered to the user group. The following is a timeline for the releases provided to the users. The project began September 2009.

TABLE 1. RELEASE TIMELINE

RELEASE	DELIVERY WEEK
Application	2
Homepage	3
Revised homepage	4
Sample project page	6
Revised sample project page	9
Pictorial from previous REU program	12
Resources/contact page	14
Delivery of completed website	16

VII. DISCUSSION

In this instance, the hybrid approach using the agile method and user-centered design for this small project was easy to implement. This section discusses the results from the study.

An exit interview with the user group revealed that they:

- Enjoyed being involved in the process
- Felt that their needs were being met
- Liked the idea of incremental releases

However, it was also noted that:

- The process was time consuming
- While there was some level of satisfaction with the progress of the project as the incremental releases were being delivered and the prototype was being used, after many weeks of meeting and seeing only a release, it was stated that it would be good "just to see the finished product"
- Confusion was also expressed with many technical aspects of the implementation

An exit interview with the development team revealed the following:

- Development was easier as they received immediate feedback from the user group
- Liked the interaction with the user group
- Appreciated the concept of pair programming

However, the team also stated:

- It was difficult to schedule meetings with the customer because of busy and conflicting schedules
- The users did not always communicate their ideas correctly which required rework of the prototype
- It was time consuming to meet for the pair programming experience due to busy and conflicting schedules as the website project was not the only project on which the individuals were working

VIII. CONCLUSION

The aim of this paper was to present the results from an empirical inquiry that focused on answering the question of how the concepts of agile methods and user-centered design could be combined to heighten user involvement in a small-scale software development project (i.e. website development). The author acknowledges that while there are many website development processes, there is inconsistency concerning the steps of the processes and many do not focus on a formalized method for actively involving the user. Consequently, the goal of the paper was to identify the broad steps involved in both agile methods, especially extreme programming, and in user-centered design and to explain how these steps could be used to create a valuable user and developer experience.

Results from the study revealed that using the agile method and user-centered design for small-scaled projects is easy to implement; however, there are certain challenges. While the user group enjoyed being a part of the process, they were overwhelmed by the involvement and certain technical aspects of development activities. Additionally,

 ISBN: 978-1-61208-117-5

the hybrid approach proved to be time consuming for both the user group and the development team.

Future work from this study includes adapting the hybrid approach to other small-scale software projects to ascertain if the type of software being developed determines the outcome of the project. Furthermore, the author intends to develop a case study specific to implementing XP and UCD.

The impact from this empirical inquiry is far reaching. It expands the dialogue that already exists among HCI researchers on how to effectively involve the user in development activities so that it is an enriched experienced. Furthermore, the study provides a foundation for future work on how light-weight software development methodologies and HCI activities can be combined for use in small-scale projects. In conclusion, as systems become more complex and user skill level decreases, it is important that designers of technology find more ways to create unique development experiences that meet both the needs of the user and the development team.

REFERENCES

[1] Internet users as percentage population. http://www.geohive.com/charts/ec_internet1.aspx (Accessed December 20, 2010).

[2] A. Dix A., J. Finlay G.B. Abowd and R. Beale. (2004). *Human-Computer Interaction*. Prentice Hall, 0130-461091, Boston, MA.

[3] Benyon, D; Davies, G; Keller, L.; Preece, J & Rogers, Y. (1998). *A Guide to Usability*, Addison Wesley, 0-201-6278-X, Reading, MA.

[4] Sharpe, H.; Rogers, Y. & Preece, J. (2007). *Interaction design: beyond human-computer interaction 2nd ed.* John Wiley & Sons Ltd, 978-0-470-01866-8, England.

[5] Preece, J.; Rogers, Y.; Sharp, H.; Benyon, D.; Holland, S. & Carey, T. (1994). *Human-Computer Interaction*. Addison Wesley, 0-201-62769-8, Reading, MA.

[6] "Computer." Def. 1. (2005). *Random House Unabridged Dictionary*. 0-375-40383-3, New York, NY.

[7] Norman, D. (1998). The Design of Everyday Things. MIT Press, Cambridge, MA.

[8] Raskin, J. (2000). *The Humane Interface.* Addison Wesley, 0-2-1-37937-6, Boston, MA.

[9] I. Sommerville. (2011). *Software Engineering 9th Ed.* Addison Wesley, 13:978-0-13-703515-1, Boston, MA.

[10] Tsui, F. and O. Karam. (2011). *Essentials of Software Engineering 2nd Ed*. Jones and Bartlett Publishers, 13:978-0-7637-8634-5.

[11] *International Standard ISO 13407 (1999).* http:// zonecours.hec.ca/documents/A2007-1-1395534. NormeISO13407.pdf. (Accessed on October 1, 2009).

[12] Schneiderman, B. 2005. Designing the User Interface 4th ed. Boston: Addison Wesley.

[13] H. Sharpe, Y. Rogers and J. Preece. (2007). *Interaction design: beyond human-computer interaction 2nd Ed.* John Wiley & Sons Ltd, 978-0-470-01866-8, England.

[14] Beck, K. (1999). Extreme programming explained: Embrace the change. Addison Wesley.

[15] Jefferies, R., A Anderson, C. Hendrickson. (2000). Extreme programming installed. In: The XP Series. Addison Wesley.

[16] Parrish, A. R. Smith, D. Hale, and J. Hale (2004). "A field study of developer pairs: Productivity impacts and implications." *IEEE Software* 21 (5), 76-9.

[17] J. Schneider and L. Johnston. (2005). "eXtreme Programming – helpful or harmful in educating undergraduates?" The Journal of Systems and Software 74, 121-132.

[18] Agile Modeling (AM) Home Page: Effective Practices for Modeling and Documentation http://www.agilemodling.com (accessed February 15, 2010).

 ISBN: 978-1-61208-117-5