SUBMISSION TO CHI 2008 WORKSHOP: OPTIMIZING AGILE UCD

Managing User-Centered Design in Agile Projects

Yael Dubinsky, Tiziana Catarci, Shah Rukh Humayoun, Stephen Kimani
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Ariosto - 25, 00185, Roma, Italy
{dubinsky, catarci, humayoun, kimani}@dis.uniroma1.it

1. Experience and Background

Our experience with agile and non-agile software teams in the industry shows low levels of User-Centered Design (UCD) management. We define UCD management for a specific software project as the ability to steer and control the UCD activities within the development environment of this project. By low levels of UCD management we mean that usually users are not involved in the process of development and if involved there is no implicitly impact in the Integrated Development Environment (IDE) that is used. Since 2002 the first author is guiding the implementation of the agile approach in both the industry and the academia [Talby et al, 2006; Dubinsky and Hazzan 2005]. Since 2006 we have integrated UCD techniques successfully in ten agile projects in the academia [Dubinsky, Catarci, and Kimani, 2007]. Based on this experience we identified the lack of UCD management as a tool in agile development environment. Since agile development environment is collaborative we claim that involving users is natural and we suggest supporting it within the IDE. Specifically, we are in the process of developing an Eclipse plug-in to support UCD management [Humayoun et al, submitted 2008].

Combining Agile and UCD

The following are the main characteristics of the integrative approach of agile and UCD that we use:

- Iterative design activities - In many cases, when user-centric techniques are used, the design of the system is refined according to the users' evaluations and this is performed mainly during the design phase. When introducing the agile approach, the design is updated regularly as the product evolved. When combining UCD activities with the agile approach, the user evaluation is fostered by performing UCD tasks in each iteration of two to four weeks, and the design is updated according to the evaluation on-going outcomes that are considered as refactoring tasks.
- Measures – Taking measurements is a basic activity in software development processes. The agile approach emphasizes it and suggests the tracker role. When combining agile and UCD, the set of evaluation tools is built and refined during the process and is used iteratively as a complement to the process and product measures.
- Roles – Different roles are defined to support software development environments. The agile approach adds roles for better management and development of the project. Combining agile and UCD adds the UCD roles, like for example the *UI Designer* role that is aforementioned.

2. For most recent Agile project, a description of the Agile team, the product the team was creating, and how UX practitioners worked with the team

In the last 2 years we have conducted several projects in the academia in which agile teams of 2-10 students[1] in each team worked according to the agile approach while involving UCD activities as part of the project development lifecycle. In this and next section we describe briefly two of these projects and some of their UCD activities [based on Dubinsky, Catarci, and Kimani, 2007].

The project goal was to develop an entrance control system that is based on the identification of faces from a set of pictures pool. The user, who wants to enter, activates the system by visual indication when he/she stands in front of a scanner, and by giving a vocal command that starts the process. The system warns the user not to do any movement before taking a series of pictures, and announces the result whether the entrance is granted or denied after comparing with the pictures in its pool.

An agile development team of eight students worked on this project and provided a deliverable of one release of three iterations of 2-4 weeks each. The project was developed during a 14-weeks semester.

With respect to the UCD activities, *thinking aloud* experiment was conducted during the second iteration. There were eleven participants who were asked to use the system, i.e., each of them asked the system to identify his/her face in order to grant the entrance. The participants were asked to report what they thought and felt during the experiment. Following is the evaluation data. Five of the participants had the following problem: they gave the vocal command but thought that it was not accepted by the system. So they started to give the vocal command again, but meanwhile the scanner took the series of pictures, the process failed, and the scanner started again due to the second command. In 2 of these 5 trials, the process ended in success, meaning the system could announce the result. Another five participants reported that using the microphone to give the vocal command enabled them with good control over the process. One participant said that he did not get any clue if to speak normally or in a louder tone.

The conclusions of the user interface designer were as follows: a) The users need an explicitly vocal or visual warning about the start and the end of the scan operation. b) The response of repeated scanning in the case of a repeated vocal command should be changed. c) The time duration in which users wait to get permission should be shortened. d) The system should tell to the user explicitly if an error occurs.

3. UCD activities that are used on Agile projects

In another project, the agile team of two students developed a speech-based mobile interface for a digital library. The interface enables vocal commands for artifact searching and to find out its location. Speech input is used for navigating the application, and speech output is also used for positioning instructions for artifact location in the physical library. The agile development team provided a product of one release of four iterations of 3-5 weeks each.

[1] The students are Computer Science major in the University of Rome "La Sapienza" and in the Technion – Israel Institute of Technology.

With respect to UCD activities, a *within experiment* was conducted with six participants who were computer science students at different levels, three males and three females. The experiment task included among other activities some searching activities. The task can be performed using speech (S) or without speech (non-S). Each of the participants performed the task in both modes S and non-S, while three participants followed S and then non-S and the remaining three followed non-S and then S. Further, before starting the experiment, each participant filled an attitude questionnaire and received ten-minute training on how to use the interface. After the experiment each of the participants filled a questionnaire to reflect on his/her activities.

An automatic time measure, which was developed as part of the system, provides us with the time stamps of each search start / end. Table 1 presents the averaged time in minutes that was invested on the two search activities by both experiment groups together with its division per mode.

Table 1. Averaged search time (in minutes)

Group	**Averaged search duration**	**Averaged Non-S search duration**	**Averaged S search duration**
Non-S → S	54.66	28	81.33
S → Non-S	26.58	14	39.16

As can be observed, the S→Non-S group performed the entire task almost twice faster than the Non-S→S group. When looking into the data of speech and non-speech per each group, we see that the participants in both groups performed the speech task slower then the non-speech task. This implies that although the speech task required more time from the participants, they learned better the system when first using it with the speech option. This knowledge was translated to specific design guidelines like for example to provide directive vocal messages as tones versus messages that it takes longer to hear and adhere to.

4. The Current Agile UCD Process

There is lack of standard ways to involve users in agile teamwork. This causes problems like the scope and effort of UCD activities that is required; in some cases, the lack of theoretical knowledge about UCD; the lack of one point of management for UCD activities which is combined with the process environment.

Our experience shows that agile management tools lack in providing the support of UCD activities [Humayoun et al, submitted 2008], and tools that support UCD activities lack in providing the management of these activities as part of the software life cycle.

Bridging this gap we present an Agile-UCD plug-in to the Eclipse platform.

5. Expectations for the Session

In this session we would like to present our lessons learned from our past experience that lead to the development of an Eclipse plug-in to support UCD management in agile development environment. We would like also to present the plug-in itself.

We manage 6 agile teams with 6 students in each for the development of the plug-in.

The stories of the first iteration of 5 weeks that are developed these days are as follows:

1. End-to-end UCD experiments:
    a. Experiments can be defined (date and time, assign users and teammates, store description, results, and conclusions)
    b. Experiments can be executed thus collecting data from User Experience (UX) according to the experiment
    c. Experiments can be viewed as per the results achieved
    d. Each group should select two kinds of experiments for implementations
2. Evaluation manager role-perspective:
    a. The evaluation manager role-perspective supports the management of the different experiments
    b. Experiments can be in different states
    c. The evaluation manager role-perspective supports
        i. Status view of the experiments
        ii. view of the current work items with respect to the experiments
        iii. UX results view
3. UI designer role-perspective:
    a. The designer role-perspective supports the UX-refactoring tasks (which are a special kind of work items) that emerge from the UX results
    b. A UX-refactoring task is associated with one or more UX results
    c. The UX results view reflects the status of the UX-refactoring tasks for example if a specific task is ‘done’ or ‘in progress’ it is marked in the UX results view
4. Work items can be created and assigned.
5. The system has one repository for its data.

By the time of this session we will have the deliverables of the second iteration of the agile teams. Our plan is to present and demonstrate the outcomes, and collect feedbacks that will be used in the third and forth iterations. Thus from one hand share with the attendees the innovation of the UCD management plug-in as part of the IDE and from the other hand enable them to participate in shaping it.

REFERENCES

Dubinsky, Y., Catarci, T., and Kimani, S. (2007). A User-based Method for Speech Interface Development, *HCI International*, Beijing, China, C. Stephanidis (Ed.): Universal Access in HCI, Part I, HCII 2007, LNCS 4554, pp. 355–364.

Dubinsky, Y. and Hazzan, O. (2005). The construction process of a framework for teaching software development methods, *Computer Science Education*, 15:4, pp. 275–296.

Humayoun, S. Dubinsky, Y. and Kimani, S. (Submitted, 2008) Integrating User-Centered Design into Agile Software Development Environments, CHI Conference, Florence, Italy.

Talby, D., Hazzan, O., Dubinsky, Y. and Keren, A. (2006) Agile software testing in a large-scale project, IEEE Software, Special Issue on Software Testing, pp. 30-37.