Eclipse Plug-in to Manage User Centered Design

Yael Dubinsky
Dipartimento di Informatica e Sistemistica "A. Ruberti"
SAPIENZA - Università di Roma
Via Ariosto - 25, 00185, Roma, Italy
dubinsky@dis.uniroma1.it

Shah Rukh Humayoun
Dipartimento di Informatica e Sistemistica "A. Ruberti"
SAPIENZA - Università di Roma
Via Ariosto - 25, 00185, Roma, Italy
humayoun@dis.uniroma1.it

Tiziana Catarci
Dipartimento di Informatica e Sistemistica "A. Ruberti"
SAPIENZA - Università di Roma
Via Ariosto - 25, 00185, Roma, Italy
catarci@dis.uniroma1.it

ABSTRACT
User-centered design (UCD) approach guides the design of user interface (UI) and its evaluation by integrating user experience as part of the software development process. Involving users during the development process by applying UCD techniques minimizes risks and increases the product quality. One of the challenges towards this is to automating the management of UCD activities during the development time thus to steer and control the UCD activities within the development environment of software projects. In this paper, we present a plug-in for Eclipse development platform to manage UCD activities at the Integrated Development Environment (IDE) level. We develop and evaluate the plug-in with teams that work according to the agile software development approach. Using this plug-in, the development teams can manage UCD activities at IDE level hence developing high quality software products with adequate level of usability.

Categories and Subject Descriptors
H5.2 [**User Interfaces**]: User-centered design, K.6.3 [**Software Management**]: Software development, software process.

General Terms
Management, Measurement, Design.

Keywords
User-centered design (UCD), user experience, agile software development, Eclipse plug-in.

1. INTRODUCTION
The user-centered design (UCD) approach [5, 15] is used to develop software products by positioning the real users of the system at the centre of design activities, e.g. by representing or modeling users in some way like scenarios and personas; through users testing of prototypes (either paper or working prototype); by involving users in making design decisions (e.g. thorough participatory design). The approach focus is on the increase of usability for the users by involving them in design and development activities. UCD activities aim at reducing the risks of the software project and increasing the overall product quality.

I-USED '08, September 24, 2004, Pisa, Italy

Variations in activities arise in different UCD methods [5, 15], and still the Human-Computer Interaction (HCI) community lacks to agree upon a precise definition of UCD methods or process [3, 8]. However, in [8] there is a set of definition of twelve principles for designing and developing systems with focus on UCD that is obtained as: *"User-centered system design (UCSD) is a process focusing on usability throughout the entire development process and further throughout the system life-cycle"* (p. 401). The International Organization for Standardization (ISO) has also defined the standard guidelines to deal with different aspects of HCI and UCD; in particular, ISO/DIS 13407[1] provides the guidance on user-oriented design process. Other relevant ISO standard guidance are ISO 9241-11[2], ISO TR 16982[3]. A detailed discussion about the methods, processes, guidelines, and prototype activities in UCD can be found in ISO standards and in [5, 15].

Lack of usability and inefficient design of the end-product are common causes amongst the others for failure of software products [12, 14]. The software products are developed for the users, and normally fail if the users find it difficult to operate them due to the lack of usability and inappropriate design. The software development teams usually work with the users either at the start of project for getting the requirements or at the end of project to test and evaluate the developed product. Furthermore, normally in the testing phase, the project teams focus more on checking the functionalities of the product (such as performance, reliability, security, robustness, etc) rather than its usability and design aspects. Checking usability or solving defects at the end of the development process needs more time, efforts, and money hence causing the failure of many software projects As a result, involving the users in the design phases is a good practice to identify lack of usability and design defects early in development time, in order to avoid any possibility of product failure at the end. The UCD approach, described above, provides methods and techniques for involving users at early stages of development [15]. So, integrating the UCD approach within software development processes gives the benefit of including the user experience as part of the development process for producing quality products with adequate level of usability.

Analyzing the current software design practices, we identified a lack of *UCD management* which we define for a specific software

[1] ISO/DIS 13407: Human Centered Design for Interactive Systems
[2] ISO 9241–11: Ergonomic requirements for office work with visual display terminals (VDTs)
[3] ISO TR 16982: Ergonomics of human-system interaction - Human-cantered lifecycle process descriptions

project as the ability to steer and control the UCD activities within the development environment of the project. This management of UCD activities at IDE level is important as it will help to integrate and automate *UCD activities* across different development life-cycle phases. By automating the management of UCD activities within development environment we decrease the time and cost to test each unit and improve the overall product quality.

In this paper, we present an Eclipse plug-in to manage the user involvement for different UCD activities in software development that can work with any software development process life-cycle. Managing UCD activities while working according to the agile software development approach [1] was already suggested [2, 4, 10, 11, 13, 7]. Our contribution is by automating UCD management at IDE level to enable, for example, creating experiments, adding users, analyzing results, and tracing the code.

The remainder of this paper is as follows. In Section 2 we describe our framework to integrate the user experience in the process of software development. Section 3 presents and explains how we can use our developed Eclipse plug-in to manage UCD activities at IDE level during software development. We conclude in Section 4.

2. USER EXPERIENCE AND THE DEVELOPMENT PROCESS

A software development approach that has been emerging in the last decade is the agile approach that is used for constructing software products in an iterative and incremental manner; where each iteration produces working artifacts that are valuable to the customers and to the project. This is performed in a highly collaborative fashion in order to produce quality products that meet the requirements in cost effective and timely manner [1].

Based on our experience with guiding the implementation of the agile approach [16, 6, 9], and the integration of UCD techniques in the last three years in agile projects in academia [7], we gather the cases in which UCD can be supported within the IDE.

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

1. **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 evolves. 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 of on-going outcomes that are considered as refactoring tasks.
2. **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.
3. **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.

Using our integrated approach of UCD and agile with software teams in the academia, we have gathered use cases to establish the plug-in specifications. Following are six representative use cases that are categorized in three themes: the development process, the evaluation activity, and the design improvement.

2.1 Development Process

There is a need to involve the users in the process of development.

Following are examples for use cases that relate to this category:

- One of the tasks during the first planning session is as follows: 'Explore who are the kinds of users who should use the product that we develop; what are their characteristics; what are their needs; what are their expectations from the product.' The customer explains that this is an important task since he cannot represent all users and actually he does not know for sure what their exact needs are (though he is sure they will like it a lot). One of the teammates asks to be assigned to this task and estimates it as 10 hours of work for this iteration. Presenting her results after two weeks, she opens her development environment in the database of the *User Perspective* and shows the list of 20 users she talked with (names, titles, contact details), main issues that were learned, and one new task that has emerged for future iterations: 'Prepare and run a questionnaire that will enable us to extract users' needs.' The customer sets high priority for this new task.
- The project manager reviews the subjects for the coming reflection session, and sees that one of the subjects is 'ways to assess the usability of our product'. She then sends invitations to seven users from the two different kinds of users to join this meeting. During the reflection session, one decision is made that two users will participate in each iteration planning session and their responsibility will be to give feedbacks on what presented. In addition they will help in defining three measures that will be automated thus enable teammates an immediate feedback during development.

2.2 Evaluation

There is a need to perform user evaluation and to manage it along the process of development.

Following are examples for use cases that relate to this category:

- The team leader browses over the details of the user experiments that are planned for tomorrow. He sees the number of users that will arrive, the names, and responsibilities of the teammates that will take care of these experiments. He checks the variables that were set and the experiments flow.
- One of the teammates sees that the *User Perspective* flushes meaning new data have arrived. He clicks on it and sees that the results of the user experiments that were conducted yesterday are in. He is surprised to find a new problem with

high severity ranking. Examining results from previous experiments, he observes that this is a new problem and adds a note about it in the discussion area. During the next iteration planning, the experiments' results are presented and among others, a measure is presented that shows two problems that emerged from the users; one in normal severity and the other one in high severity.

2.3 Design Improvement

There is a need to improve the design of the user interfaces based on the evaluation results.

Following are examples for use cases that relate to this category:

- The designer of the user interface views the latest design diagrams and tries different changes that adhere to the new task in this iteration. The task was added due to the last problem that was found by the users. Thinking of different options, she talks with two users and receives their feedbacks. She shows them the possible drawings of the new interface and asks them to simulate trying it while thinking aloud. She summarizes the results and sets her decision.
- One of the teammates browses over the system reports and looks for each user experiment, which was conducted in the last two releases, what were the results and what were the implications on design. For each implication, he sees the development tasks that are related.

We suggest that the combination of the agile and UCD approaches should be supported by an extension to a contemporary development environment in order to be used in a natural manner. This is elaborated in the next section.

3. THE UCD MANAGEMENT PLUG-IN

3.1 The Project

A team of six developers in a project based course in the academia has developed the UCD plug-in that is presented in this paper[4]. The project took five and a half months and was composed of 4 iterations, three of 5 weeks each and one of 3 weeks. Table 1 shows the durations and the main themes of each iteration.

Table 1. The iterations – duration and themes

Iter.	Duration (weeks)	Themes
1	5	Experiments and Roles - End-to-end experiments: define the experiment, execute it, results view - Evaluation manager role-perspective - UI designer role-perspective - Work items can be created, assigned - The system has one data repository
2	5	Users' interface and user experience (UX) automation - Users' management and permissions - A user interface to run the experiment - Client / Server architecture for running the experiment - Support automatic measures for user evaluation that are derived from user experience - On line help - Traceability – experiments should be part of a specific project that we develop; each specific development task that is derived from one or more experiments results should be associated with the appropriate code parts that implement them
3	3	Stability - Testing and Refactoring - Development refinement
4	5	Heuristics and User Profiling - Support Nielsen heuristics technique - Support user profiling - Scale with more *end* projects

[4] This project was developed as part of the "Annual Project in Software Engineering" course that is instructed by the first author at the Computer Science Department at Technion IIT.

3.2 Using the UCD Management Plug-in

The main feature of the UCD management plug-in is the ability to create and deploy user experiments from within the Eclipse IDE. Focusing on a specific software project, we can define different kinds of experiments. One kind for example is a task-based user experiment in which the participant uses the target product and receives the tasks to perform along the experiment. During this experiment the system measures different performance times. Another kind of experiment is questionnaire-based experiment in which the user specifies the level of his/her agreement with the presented set of statements. The development team chooses the set of experiments according to the nature of software project and then selects appropriate users from the pool of target users to perform these tasks.

We illustrate the definition of a task-based user experiment using a view that is presented by Figures 1 and 2.

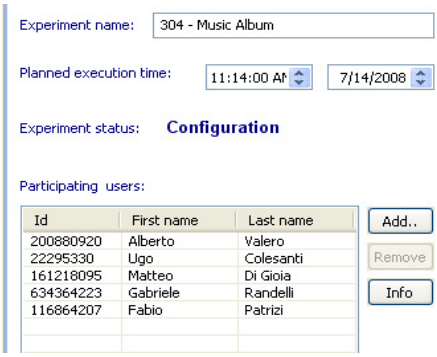

Figure 1. Defining the experiment (left hand side)

In the left hand side of the view (Figure 1) we can see the options of setting the experiment schedule and the users who are involved. In the right hand side (Figure 2) we can see the options of adding tasks to the experiment, save the experiment, and execute it.

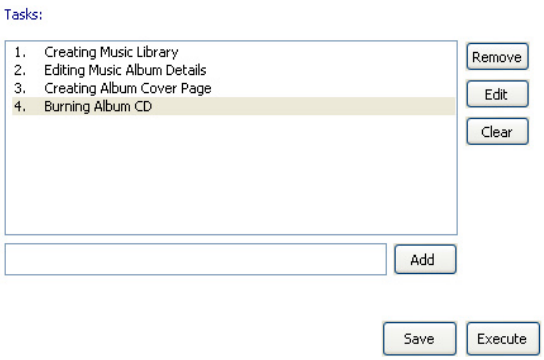

Figure 2. Defining the experiment (right hand side)

The experiment can run locally i.e., on the server on which the data is stored, or remotely. Setting the remote option causes the enlisted users to receive email with the experiment files attached, so they can perform the experiment in a way that the results are stored in the server. Figure 3 shows the results view of a specific experiment. Different kinds of experiments were developed that support appropriate results views.

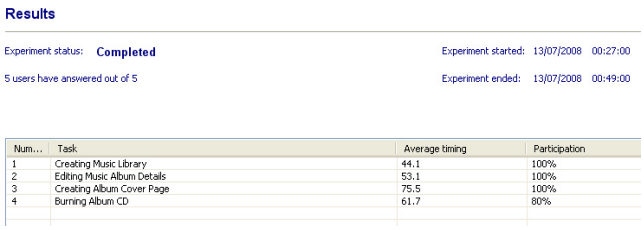

Figure 3. The experiment's results view

Experiment Explorer is available to support the experiments of a specific project (Figure 4). Experiments can be shared among different projects.

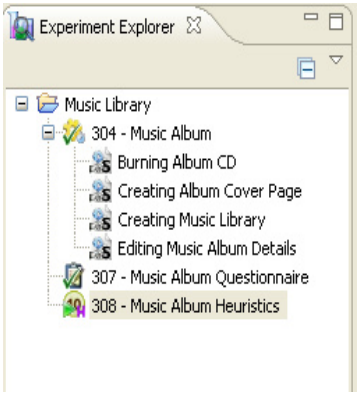

Figure 4. The Expriement Explorer

Managing the experiments, new development tasks are derived. These tasks are the results of the already conducted experiments. The plug-in enables associating code part/s to the appropriate task/s and vice versa so traceability is kept. Figure 5 shows how a code segment can be associated.

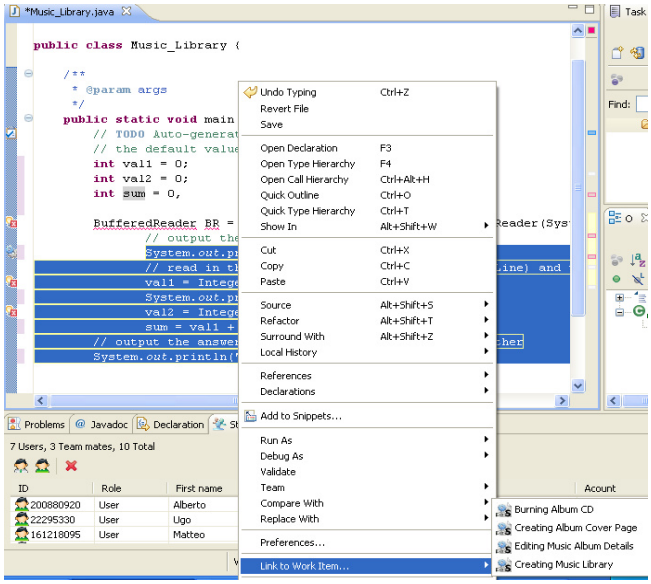

Figure 5. Associating code to a development task

Figure 6 shows how this code is marked (left side bar) and highlighted.

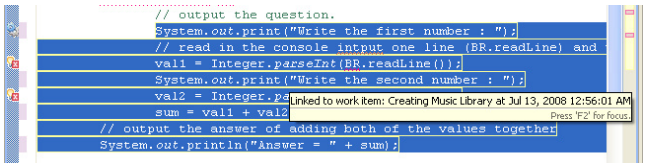

Figure 6. Associated code is marked

3.3 Evaluating the Plug-in

As part of the third iteration, the team was asked to evaluate its own product (the UCD management plug-in) using itself ("eating own cookies"). Following is the plan and the results of this preliminary evaluation.

The evaluations goals as written by the team were:

- Examining suspicious issues like adding new users to the system and analyzing the experiments' results (specifically for the questionnaire-based experiments).
- Receiving feedback on the graphical user interface (GUI) and how intuitive it is.
- Examining the plug-in on a large scale project.

Two kinds of experiments were defined by the team for the evaluation of the plug-in. The first experiment was a task-based experiment and the second was a questionnaire-based experiment.

The participants were 3 students from another team in the same course, and in addition all the six developers performed both kinds of experiments. Each participant performed the experiment by himself / herself while one observer was sitting aside for writing notes.

Results

Experiment status: **Running**

3 users have answered out of 6

Experiment started: 17/03/2008 11:32:00

Experiment ended: ---

		Strongly Disagree	Disagree	Agree	Strongly Agree
1.	Logging in to the system is simple.	0	0	2	1
2.	Adding a user or a teammate to the system is simple	0	1	0	2
3.	Switching between teammates is fast and simple.	1	2	0	0
4.	The configuration page is intuitive.	0	0	2	1
5.	The Questionnaire result page displays the level of agreement (p...	0	0	3	0
6.	The Questionnaire result page displays the usability problems disc...	0	2	1	0
7.	The different editors and views of the plug-in are uniform and foll...	0	0	1	2
8.	The different editors and views of the plug-in blend seamlessly in...	0	1	2	0
9.	I would use this plug-in to test the usability of an application in de...	0	0	1	2

Figure 7. Results of questionnaire-based experiment – participants from another team

Results

Experiment status: **Completed**

6 users have answered out of 6

Experiment started: 17/03/2008 10:56:00

Experiment ended: 17/03/2008 13:39:00

		Strongly Disagree	Disagree	Agree	Strongly Agree
1.	Logging in to the system is simple.	0	0	0	6
2.	Adding a user or a teammate to the system is simple.	1	3	1	1
3.	Switching between teammates is fast and simple.	3	2	0	1
4.	The configuration page is intuitive.	0	0	2	4
5.	The Questionnaire result page displays the level of agreement (p...	0	0	1	5
6.	The Questionnaire result page displays the usability problems disc...	3	2	0	1
7.	The different editors and views of the plug-in are uniform and foll...	0	1	4	1
8.	The different editors and views of the plug-in blend seamlessly in...	0	0	3	3
9.	I would use this plug-in to test the usability of an application in de...	0	1	2	3

Figure 8. Results of questionnaire-based experiment – team members are the participants

We focus on the questionnaire-based experiment and comments of the observers and show an example of a derived task that emerged for further development. The questionnaire included the following statements:

1. Logging in to the system is simple.
2. Adding a user or a teammate to the system is simple.
3. Switching between teammates is fast and simple.
4. The configuration page is intuitive.
5. The Questionnaire result page displays the level of agreement (per statement) in a clear way.
6. The Questionnaire result page displays the usability problems discovered in a clear way.
7. The different editors and views of the plug-in are uniform and follow a similar theme
8. The different editors and views of the plug-in blend seamlessly into the eclipse.
9. I would use this plug-in to test the usability of an application in development.

Figures 7 and 8 show the results of the three participants from another group and the results of the developers themselves respectively.

Following are few comments, for example, that were presented by the observers:

1. "In the questionnaire-view that is presented to the participant, long tasks appear truncated."
2. "The participant did not know how to save the changes in the result page. He searches for a save button like appears in other screens."
3. "The names of the operations in the menu of the experiment view are not clear."

Analyzing the results of both experiments, associations to the specific results were presented for each conclusion, and then suggested development tasks were associated to the conclusions.

One of the finding, for example, was detailed as follows:

"It was found that there is a difficulty in identifying problems in the product out of the information that is presented in the 'results page'. Participants find it hard to associate the results (as presented in the 'results page') to the experiment goals and to the practical problems that were discovered."

"Association to the results:

- In the questionnaire-based experiment the two teams marked 'Disagree' for statement 6 [The questionnaire result page displays the usability problems discovered in a clear way].
- In the task-assignments experiment, it took long time, 84 and 177 seconds in average for the two groups, to complete task 5 [According to the experiment goals, try to assess the

number of usability problems indicated by the results, and write that number as a conclusion to this experiment]."

The development task that was defined using the plug-in is as follows: "Enable determining thresholds for success and failure in an experiment and present them clearly in the 'results page'."

4. CONCLUSION

In this paper, we present our Eclipse plug-in to automating the process of managing UCD activities at the Integrated Development Environment (IDE) level during the development time of software projects. To develop the framework we were inspired by use cases that emerged when performing UCD activities with the agile teams. Using this plug-in, the software project team can create experiments, adding users, analyzing results and tracing back it to code for their developed or in-progress product. By automating the process of managing UCD activities the chances of creating quality products with adequate level of usability become high, as it helps to get benefits of user experience during development time.

In future, we intend to continue work on the developed plug-in to manage more UCD activities. Further, we also intend to evaluate the developed product on big scale with different size of software development teams.

5. ACKNOWLEDGMENTS

Our thanks to the plug-in developers from Technion IIT whose product is presented in this paper: David Ben-David, Tomer Einav, Yoav Haimovitch, Barak Nirenberg, Laliv Pele, and Alon Vinkov.

6. REFERENCES

[1] Agile Alliance 2001. Manifesto for Agile Software Development. Technical Report by Agile Alliance, http://www.agilealliance.org.

[2] Blomkvist, S. 2005. Towards a Model for Bridging Agile Development and User-Centered Design. Published as a book chapter: Seffah, A., Gulliksen, J., and Desmarais, M., (eds.). Human-Centered Software Engineering – Integrating Usability in The Development Process. Springer, Dordrecht, The Netherlands, 217-243.

[3] Blomkvist, S. 2006. User-Centered Design and Agile Development of IT Systems. IT Licentiate theses, Department of Information Technology, Uppsala University.

[4] Detweiler, M. 2007. Managing UCD within Agile Projects. ACM Interactions May-June, 40 – 42.

[5] Dix, A., Finlay, J.E., Abowd, G.D., and Beale, R. 2003. Human Computer Interaction, 3rd Edition, Prentice Hall.

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

[7] Dubinsky, Y., Catarci, T., Humayoun, S., and Kimani, S. 2007. Integrating user evaluation into software development environments, 2nd DELOS Conference on Digital Libraries, Pisa, Italy.

[8] Gulliksen, J., Goransson, B., Boivie, I., Blomkvist, S., Persson, J. and Cajander, A. 2003. Key principles for user-centered systems design. Behaviou & Information Technology, Vol. 22, No. 6, 397–409.

[9] Hazzan, O. and Dubinsky, Y., Agile Software Engineering, Undergraduate Topics in Computer Science Series, Springer-Verlag London Ltd, 2008, in press.

[10] Hudson, W. 2003. Adopting User-Centered Design within an Agile Process: A Conversation. Cutter IT Journal, (16), 10 http://www.suntagm.co.uk/design/articles/ucdxp03.pdf

[11] Hwong, B., Laurance, D., Rudorfer, A., and Schweizer, A. 2004. User-Centered Design and Agile Software Development Processes. Siemens Corporate Research http://www.scr.siemens.com/en/pdf/se_pdf/rudorfer-1.pdf.

[12] Landauer, T. K. 1995. The trouble with computers: usefulness, usability, and productivity, MIT Press.

[13] McInerney, P., and Maurer, F. 2005. UCD in agile projects: Dream team or odd couple?. ACM Interactions, 12(6), 19 - 23.

[14] Norman, D. 2006. Why Doing User Observations First Is Wrong, ACM Interactions, July-August 2006.

[15] Sharp, H., Rogers, Y., and Preece, J. 2007. Interaction Design: Beyond Human-Computer Interaction. 2nd Edition. Willey.

[16] 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, 30-37.