UEMan: A Tool to Manage User Evaluation in Development Environments

Shah Rukh Humayoun
Dipartimento di Informatica e Sistemistica "A. Ruberti"
Sapienza - Università di Roma
humayoun@dis.uniroma1.it

Yael Dubinsky*
IBM Haifa Research Lab
Mount Carmel, Haifa 31905
dubinsky@il.ibm.com

Tiziana Catarci
Dipartimento di Informatica e Sistemistica "A. Ruberti"
Sapienza - Università di Roma
catarci@dis.uniroma1.it

Abstract

One of the challenges in software development is to collect and analyze the end users' feedback in an effective and efficient manner. In this paper we present a tool to manage user evaluation alongside the process of software development. The tool is based on the idea that user evaluation should be managed iteratively from within the integrated development environment (IDE) in order to provide high quality user interface. The main capabilities include creating the experiment object as part of the software project; deriving development tasks from the analysis of evaluation data; and tracing these tasks to and from the code. Further, we provide a library to enable development of Java aspects for creation of automatic measures to increase the body of the evaluation data. Using this tool, development teams can manage user-centered design (UCD) activities at the IDE level, hence developing software products with an adequate level of usability.

1. Introduction

Software development needs to be continuously contributed by the end users for whom the software is developed. Even though it is easy to agree on this point, no existing tools support it as part of the integrated development environment (IDE).

To bridge this gap, we present a user evaluation manager (UEMan) tool, which is an Eclipse plug-in that supports the management of user evaluation as part of the Eclipse IDE. UEMan enables the definition and deployment of experiments to be performed by end users and by usability experts. Based on the evaluation data that is gathered by these experiments, the user-centered design (UCD) [1, 4, 7] evolves, and UCD tasks are added to improve the code accordingly. Traceability is maintained in order to encourage correctness and follow-up between code changes and the evaluation results.

Our research approach involves working with software teams to understand how UCD can be embedded in the software development process and contribute towards producing high quality software products. This understanding helps us to shape a set of requirements for tooling as well as guidelines and techniques to accompany it [2]. The research includes developing UEMan, validating the concept of UCD integration within the tools of development, and evaluating it with software teams [3].

In Section 2 we describe the concept of automating user evaluation in the IDE. In Section 3 we present UEMan and provide evaluation data. We conclude in Section 4.

2. Automating user evaluation within the development process

A lack of usability and inefficient design of the end-product are common causes, amongst others, for failure of software products [5, 6]. Users are either involved at the beginning of projects with defining the requirements or at the end of project with testing and evaluating the developing product. In the testing phase, the project teams focus more on checking the functionalities of the product 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 they are not usually performed.

The UCD approach provides techniques to involve users at the early stages of development [7]. Automating these techniques within the IDE, as described in what follows, provides the on-going benefit of including the user experience as part of the development process for producing high quality products with an adequate level of usability.

* This work is done as part of a visiting membership at the Human Computer Interaction Group at the Dipartimento di Informatica e Sistemistica "A. Ruberti", Sapienza Università di Roma.

2.1 The experiment object

To help end-users continuously evaluate the product, the UCD approach [1, 4, 7] provides different kinds of tests, following referred to as *experiments*. UEMan supports four kinds of experiments[2]: *task-based* experiments to judge the usability level of developing product by performing different tasks by the end-users; *questionnaire-based* experiments to evaluate the system by the level of agreement with the presented statements; *logging-aspect* experiments to record end-users' behavior; and *Nielsen-heuristics* experiments to judge usability by experts. Automating the experiments means that in the development area of a software project we can add a new kind of an object named *experiment* that can be created and executed to provide evaluation data. Further, the experiment's results can be associated with future development tasks that emerge to answer the issues these results raise.

Derived UCD tasks. Each kind of evaluation experiment has its own criteria for judging the usability level of the product. Support for the analysis of the experiments' results enables comparison of the results against the targeted usability standards. If the results show a failure to achieve the target usability level then new development tasks can be defined accordingly. Each task is associated with the relevant data thus provides its rationale with one mouse click.

Code traceability. The product evolves through iterative steps in which the design of the software project improves gradually. Automating the process of keeping backward and forward traceability between different evolving parts at the IDE level gives a better understanding of the refinement done in the design to improve the product. The traceability is maintained through associations between the code parts, experiments, and derived development tasks. This mechanism helps among others to learn about the evaluation impact.

2.2 Developing evaluation Java aspects

Another advantage of introducing automated UCD techniques into the IDE is enabling the developers to add automatic evaluation hooks in the software under development. Using a Java library, the developers can create and implement Java aspects that are customized for the specific software. For example, an aspect can be created to control the use of a specific button or key that is part of the developing software. Running an experiment that includes such a measure provides insights about the users' behavior.

[2] In this paper we present the first three kinds of experiments.

3. The UEMan Tool

UEMan is an Eclipse [8] plug-in, developed in the Eclipse IDE using its Plugin Development Environment (PDE) facility to extend and become an integral part of the Eclipse IDE.

3.1 Main features of UEMan

UEMan provides an Experiment Explorer that helps to create, manipulate, and automate evaluation experiments for a specific project (Figure 1). It also facilitates sharing these evaluation experiments among different projects, to get the benefits from the evaluation results data of one project into other projects.

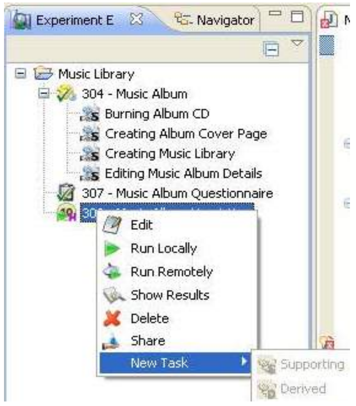
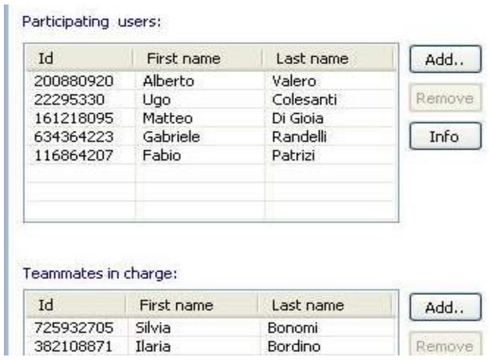


Figure 1. The Experiment Explorer

Using the explorer, one can run the experiment either locally i.e., on the server on which the data is stored; or remotely, in which the enlisted users receive an email with the experiment files attached and instructions to run the experiment in such a way that the results are stored back on the server.


Participating users:

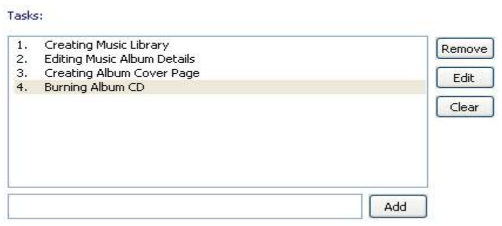
Id	First name	Last name
200880920	Alberto	Valero
22295330	Ugo	Colesanti
161218095	Matteo	Di Gioia
634364223	Gabriele	Randelli
116864207	Fabio	Patrizi

Teammates in charge:

Id	First name	Last name
725932705	Silvia	Bonomi
382108871	Ilaria	Bordino

Figure 2. Configuring UEMan users

Configuring an experiment is performed using the experiment's Configuration Wizard. Figure 2 (the left-hand side of the wizard) shows the option for adding participating end-users and teammates responsible for the evaluation experiment. Figure 3 (the right-hand side) shows the list of tasks the participants will perform while executing a task-based type of evaluation experiment.

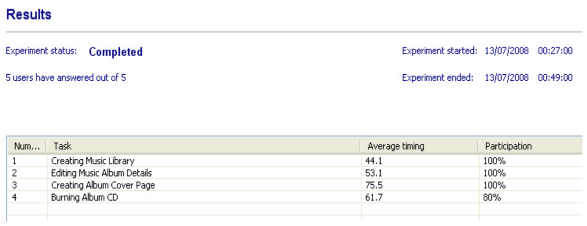


Tasks:

1. Creating Music Library
2. Editing Music Album Details
3. Creating Album Cover Page
4. Burning Album CD

Figure 3. Configuring experiment tasks

While a user performs the experiment, UEMan measures different performance times to evaluate the usability level of the related parts of the software product. Figure 4 shows the results view of a task-based experiment. You can see the average time (in seconds) and the level of users' participation in performing each task.



Results

Experiment status: Completed

5 users have answered out of 5

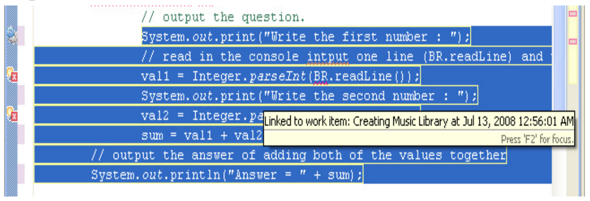
Experiment started: 13/07/2008 00:27:00

Experiment ended: 13/07/2008 00:49:00

Num...	Task	Average timing	Participation
1	Creating Music Library	44.1	100%
2	Editing Music Album Details	53.1	100%
3	Creating Album Cover Page	75.5	100%
4	Burning Album CD	61.7	80%

Figure 4. The experiment's results view

UEMan enables associating a code file or code parts with the appropriate experiments or tasks within the experiment or to any created development tasks, and vice versa, for the purpose of traceability. Figure 5 shows a code part that is marked on the left-side bar and highlighted to show the association with a specific experiment.



```
            // output the question.
            System.out.print("Write the first number : ");
            // read in the console intput one line (BR.readLine) and
            val1 = Integer.parseInt(BR.readLine());
            System.out.print("Write the second number : ");
            val2 = Integer.pa
            sum = val1 + val2
        // output the answer of adding both of the values together
        System.out.println("Answer = " + sum);
```

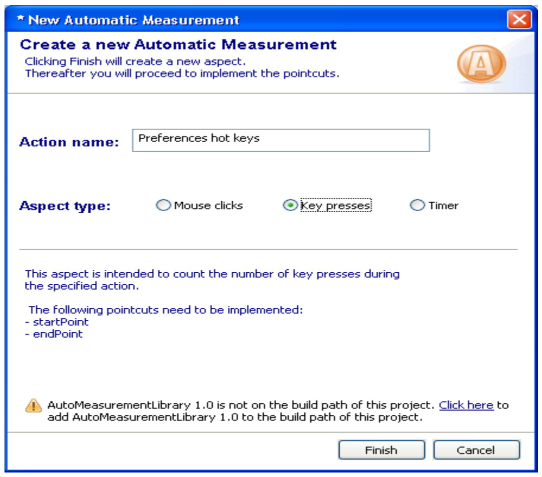
Linked to work item: Creating Music Library at Jul 13, 2008 12:56:01 AM

Press 'F2' for focus.

Figure 5. Associated code is marked

3.2 Automatic measures by Java aspects

UEMan provides a Java library that enables adding Java aspects to the software under development to support automatic measures that fit the developing product. These measures, as part of a logging-aspect experiment, record different kinds of user behavior while using the evaluated software product. Figure 6 shows the creation wizard, including the list of aspect's pointcuts that should be implemented.



New Automatic Measurement

Create a new Automatic Measurement

Clicking Finish will create a new aspect.
Thereafter you will proceed to implement the pointcuts.

Action name: Preferences hot keys

Aspect type: Mouse clicks [✓ Key presses] Timer

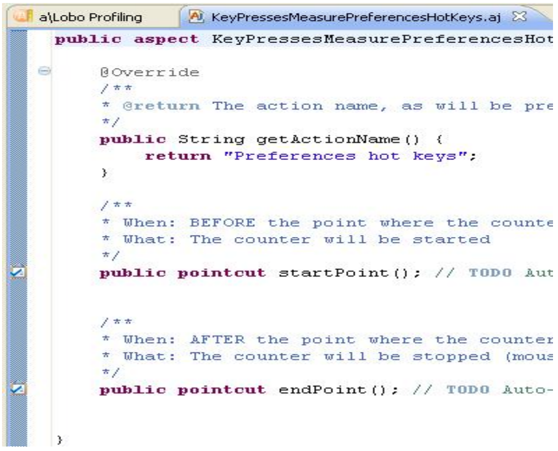
This aspect is intended to count the number of key presses during the specified action.

The following pointcuts need to be implemented:
- startPoint
- endPoint

AutoMeasurementLibrary 1.0 is not on the build path of this project. Click here to add AutoMeasurementLibrary 1.0 to the build path of this project.

Figure 6. Creating an aspect

Figure 7 shows the created aspect in code and the pointcuts to implement. Using the *AutoMeasurement* Library, the development team can analyze automatically recorded evaluation data to assist with the future design of the product.



```
public aspect KeyPressesMeasurePreferencesHot

    @Override
    /**
    * @return The action name, as will be pre
    */
    public String getActionName() {
        return "Preferences hot keys";
    }

    /**
    * When: BEFORE the point where the counte
    * What: The counter will be started
    */
    public pointcut startPoint(); // TODO Aut


    /**
    * When: AFTER the point where the counter
    * What: The counter will be stopped (mous
    */
    public pointcut endPoint(); // TODO Auto-


}
```

Figure 7. Implementing the pointcuts

The recorded data can be viewed as text or using visual graphs. For example, see in Figure 8 the UEMan graphical view of the results of a timer that measures the time spent in different windows when evaluating the Lobo Java web browser[3]. This kind of logging-aspect experiment provides a new, deeper level of user involvement and shows how UCD activities can be incorporated into software development.

[3] Lobo web browser at http://lobobrowser.org/.

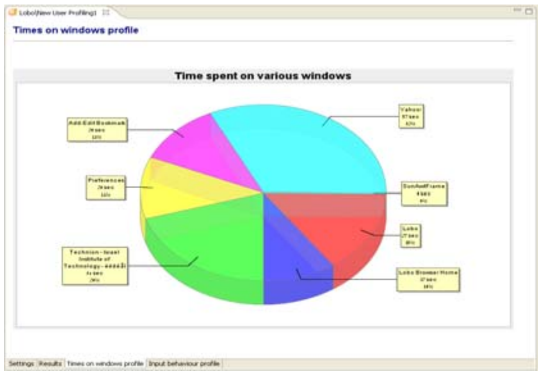
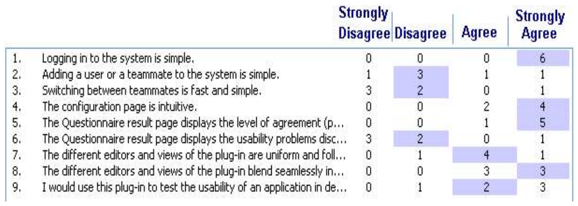


Figure 8. Evaluating Lobo using UEMan

3.3 Evaluating UEMan

As part of UEMan development, the team was asked to evaluate its own product using itself ("eating its own cookies"). The team defined two experiments: a task-based experiment and a questionnaire-based one. Nine participants performed the experiments while an observer sat alongside, writing notes.

The questionnaire-based experiment included statements that were related to the UEMan user interface, e.g., "The Questionnaire results page clearly displays the usability problems discovered". The results view as shown in Figure 9 shows the results after six participants have answered. The view presents the level of agreement for the different statements. The sixth line is the statement we gave as an example, for which we can see there is general disagreement (2 disagree; 3 strongly disagree; 1 strongly agrees).



		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 9. Results view of questionnaire-based experiment

Using triangulation of evaluation experiments, another kind of data was observed from the task-based experiment that highlighted the same problem. One of the tasks asked to assess the general level of usability indicated by the results. The idea was to see if the user can easily conclude upon viewing the results. It happened that it took 146 seconds in average to complete this task and the team saw this as long time.

As a result of this data and analysis, the following development task was derived: "Enable determining thresholds for success and failure in an experiment and present them clearly in the 'results page'". The actions that followed were: the task was assigned to one of the team members; the relevant code was developed and associated to the experiment result.

4. Conclusion

In this paper we presented the UEMan tool, an Eclipse plug-in, for automating the process of managing UCD activities at the IDE level. The automation process helps the project team be more efficient and effective in receiving users' evaluation feedback and improve the design of the developing product; thus improving the overall product quality and decreasing the cost, time investment, and total effort. Using UEMan, the software project teams can create evaluation experiments, add users, analyze results, and trace it back to the code for their developed or in-progress product.

In the future, we intend to continue work on UEMan to automate more UCD activities and add components to provide statistical analysis to the appropriate kinds of experiments. Further, we intend to continuously evaluate UEMan by using it to evaluate various software development projects.

5. Acknowledgments

Our thanks go to the developers from Technion Israel Institute of Technology: David Ben-David, Tomer Einav, Yoav Haimovitch, Barak Nirenberg, Laliv Pele, and Alon Vinkov.

6. References

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

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

[3] Dubinsky, Y., Humayoun, S. R., and Catarci, T., Eclipse Plug-in to Manage User Centered Design, Workshop on the Interplay between Usability Evaluation and Software Development (I-USED), 2008.

[4] 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.

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

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

[7] Sharp, H., Rogers, Y., Preece, J. 2007. Interaction Design: Beyond Human-Computer Interaction.Willey.

[8] The Eclipse Platform: http://www.eclipse.org , The Eclipse Foundation, 2008.