A Three-Fold Integration Framework to Incorporate User–Centered Design into Agile Software Development

Shah Rukh Humayoun[1], Yael Dubinsky[2], and Tiziana Catarci[1]

[1] Dipartimento di Informatica e Sistemistica "A. Ruberti",
SAPIENZA - Università di Roma, Via Ariosto – 25, 00185, Roma, Italy
[2] IBM Research – Haifa, Mount Carmel, Haifa 31905, Israel
{humayoun,catarci}@dis.uniroma1.it,
dubinsky@il.ibm.com

Abstract. We present a framework that incorporates user-centered design (UCD) philosophy into agile software development through a three-fold integration approach: at the process life-cycle level for the selection and application of appropriate UCD methods and techniques in the right places at the right times; at the iteration level for integrating UCD concepts, roles, and activities during each agile development iteration planning; and at the development-environment level for managing and automating the sets of UCD activities through automated tools support. We also present two automated tools—UEMan and TaMUlator, which provide the realization of the development-environment level integration.

Keywords: User-centered design (UCD), agile software development, usability evaluation, integrated development environment (IDE), UEMan, TaMUlator.

1 Introduction

One of the challenges in software development is to involve end users in the design and development stages so as to collect and analyze their behavior and feedback in an effective and efficient manner and then to manage the ensuing development accordingly. One way to achieve this is by applying user-centered design (UCD) [4] philosophy. This philosophy puts the end users of the system at the centre of the design and evaluation activities, through a number of methods and techniques. UCD is applied in software projects with the aims of increasing product usability, reducing the risks of failure, decreasing long-term costs, and increasing overall quality. Integrating UCD activities into software development processes fuses the user experience with the development process, attaining a high level of usability in the resulting product. One description of UCD that we find particularly motivating is: "*User-centered system design (UCSD) is a process focusing on usability throughout the entire development process and further throughout the system life-cycle*" [10] (p. 401).

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

M. Kurosu (Ed.): Human Centered Design, HCII 2011, LNCS 6776, pp. 55–64, 2011.

Generally, in software development practice, software teams hesitate to imply UCD activities due to their time-consuming and effort-intense nature. Using the agile approach, in which customers and product owners lead the prioritization of the development, helps developers overcome these hesitations. By emphasizing the benefits common to both the end users and the developers, UCD and the agile approach can be dynamically integrated to get benefits from both, resulting in the development of high-quality and usable software products.

We have been working with agile and non-agile software development teams in industry and in academia [5, 11, 17] for several years, and we have successfully integrated parts of UCD philosophy into agile development environment [6, 7, 12]. Based on this experience, we identified that agile development teams were often lacking a properly-integrated approach that utilizes the UCD philosophy from end-to-end at all levels. To overcome this gap, we propose a three-fold integration framework that gives suggestions and recommendations for involving UCD in agile software development at different levels. Our approach identifies ways to apply appropriate UCD activities alongside agile development activities, with the aim of developing high-quality and usable products.

The remainder of this paper is structured as follows: In Section 2, we describe other approaches that also integrate UCD into software development processes. In Section 3, we describe our three-fold integration approach in detail. In Section 4, we present two automated tools—UEMan and TaMUlator, which provide realization of parts of our framework at the integrated development environment (IDE) level. In Section 5 we briefly describe two case studies in which development teams used our development-environment level integration approach for evaluating their software projects along development. Finally, we present our conclusion and suggest future directions in Section 6.

2 Related Work

Integrating UCD philosophy into software development processes, and specifically into the agile development approach, is not a new idea. Some past works focus on applying several UCD techniques to agile development, while others focus on the benefits a particular technique can bring to agile development. However, the literature lacks a consolidated approach to cover the integration from all aspects.

Göransson et al. [9] defined a usability design process that integrates their previously defined user-centered system design (UCSD) [10] approach with software development processes. Their defined process is iterative-in-nature and works with well-planned iterative and incremental development approaches, such as one provided for Rational Unified Process (RUP). Their process is divided into three phases: *requirements analysis*, *growing software with iterative design*, and *deployment*. Their approach is not very well suited for agile development, in which emphasis is given to the working artifacts and small iterations.

Chamberlain et al. [3] described a framework for integrating UCD techniques into agile development and provided a field study. They identified a set of five principles as

being significant in integrating the two approaches—user involvement, collaboration and culture, prototyping, project lifecycle, and project management.

Sy [16] described the use of cycle-zero in his integrating approach. Pattern [15] proposed using of interaction design for agile development through Constantine and Lockwood's usage centered design approach. Hussain et al. in [13] proposed an approach of integrating UCD and XP development based on evaluating the usability of user interfaces of developing application in small iterative steps. Fox et al. provided a study [8] that researched participants experienced in combining these two approaches and concluded that there can be a common model from the existing models.

3 The Three-Fold Integration Framework

The UCD and agile development processes differ in nature. The two approaches were developed in different environments and disciplines, but can be integrated if care is properly taken, as their basic concepts and philosophies have no fundamental contradictions. Our approach emphasizes a tight integration from top-to-bottom, in which shared ideas are combined at every level, from the process life cycle to the development environment, gaining the benefits of both approaches. Our three-fold integration framework incorporates UCD into agile development at three levels: the process life-cycle level, the iteration level, and the development-environment level. The following sub-sections describe each of these levels.

3.1 The Life-Cycle Level Integration—Selection and Application of UCD Methods and Techniques in the Agile Process Life Cycle

We define life-cycle level integration as performing appropriate UCD methods and techniques in the right places at the right times, alongside the other development tasks. We distinguished the different types of UCD methods into two groups, elicitation and evaluation, based both on the way the methods perform and on their impact on software project development.

Elicitation Methods: These UCD methods are used for eliciting requirements and design of the software project. Normally, the end users or UCD experts are involved during the initial phases of development lifecycle. We suggest using these in early activities of agile development iterations to give more attention to eliciting requirements and design. Among the different elicitation methods, those that take less time, efforts, and give high feedbacks, such as focus groups and card sort methods, are more suitable as they fit perfectly in agile development.

Evaluation Methods: These UCD methods involve end users, UCD experts, and automated tools and are used to evaluate developing/developed products by identifying usability issues. Each type of evaluation method highlights only parts of usability issues, and only to a certain extent, so using more than one method is recommended [14] for covering a higher rate of usability issues. The evaluation methods performed by UCD experts are useful in early design activities, in which only paper prototypes or only parts

of working prototypes are available, as these methods take less time and effort, both of which are at a premium during these early activities of development iteration. On the other hand, evaluation methods performed by end users give better results when they are used on working prototypes, as these prototypes help end users understand the system, taking better advantage of the methods. Heuristic evaluation, question-asking protocol, and performance measurement [4, 14] are all examples of evaluation methods.

Our framework provides a set of attributes for selecting appropriate elicitation and evaluation methods to apply during agile development iteration activities. Table 1 shows these attributes and describes each one from the agile development perspective, i.e., short-time iterations, high-level of collaboration with customers, focus on working artifacts, and dynamic processes.

Table 1. Set of attributes for determining the selection of appropriate UCD methods

Attribute	Description
Automation	The automated tools support and the level of automation (e.g., *None*, *Capture*, *Analysis*, *Critique* (taxonomy by Balbo [2])
Effectiveness	Feedback effects (*Low*, *Medium*, *High*) on design or development
Dynamicity	The ability of the method to be changed according to the target environment (*Low*, *Medium*, *High*)
Time-cost	How much minimum time is needed to complete this method
Effort-cost	How much efforts are needed to perform this method (e.g., man power, equipments, experiment place, other resources)
Ease of Learning	How easy it is to learn the method, both for responsible persons on the development team and/or for the end users who will perform it
Results Accuracy	Accuracy of results
Covering Area	The usability issues covered by a method

Along these attributes, selecting an appropriate UCD method also depends on other factors, such as life-cycle stage, availability of evaluators, etc. For early design activities, we recommend an emphasis on paper-based or simple UI prototype-based evaluation methods to improve design early. While in later iteration activities, we recommend using formal evaluation methods to get formal results. We also recommend using a mixture of evaluation methods, preferably supported by automated tools and performed by end users and UCD experts for maximum results. Automation tools support gives more accurate results and save time and costs—all factors that complement agile development.

On the basis of the above recommendations, we suggest a life cycle of four UCD activities for involving UCD philosophy alongside the agile development iteration. Figure 1 shows the UCD activities (solid ovals), in which agile activities that are done per each user story or development task are represented by dashed lines ovals. Sometimes a UCD activity overlaps one or more agile activities.

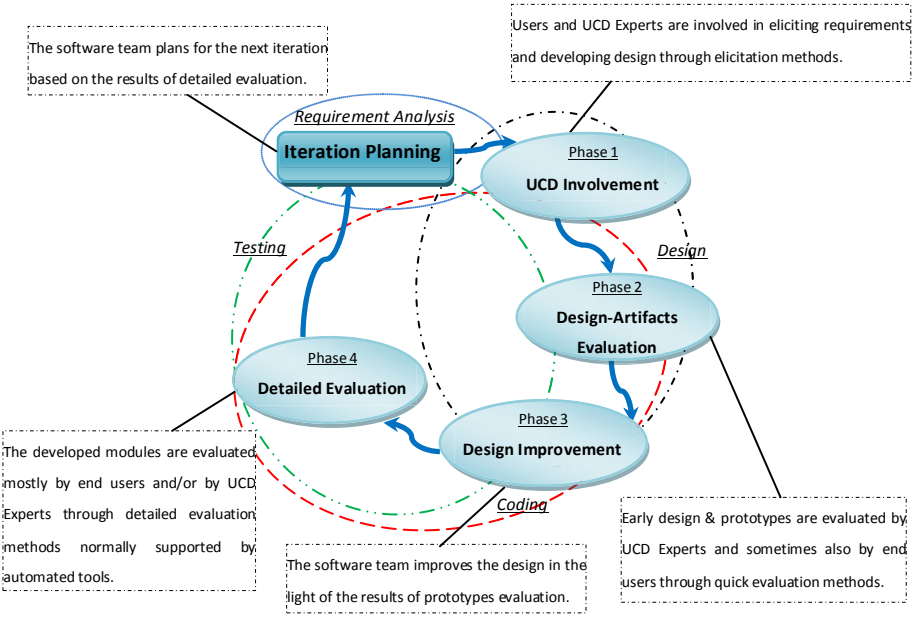

Fig. 1. Life cycle for involving UCD philosophy into agile development iteration

UCD Involvement: The software team involves end users and UCD experts when appropriate, mostly through the use of elicitation methods during work on requirements, design, and early prototypes.
Design-Artifacts Evaluation: The early design and prototypes are then quickly evaluated through short-time consuming evaluation methods normally by UCD experts (i.e., system analysts, usability evaluators, UI designers) and sometimes by a small group of end users. The results of this phase become input for the third phase.
Design Improvement: The software team corrects and improves the design according to the feedback and performs implementation of the target modules.
Detailed Evaluation: The developed modules are evaluated in detail by end users and/or by UCD experts, normally through rigid evaluation methods with automated tools support. The results, feedback, and suggestions serve as input for making plans for improvements in the design and for the implementation of developing products in the upcoming iterations.

3.2 The Iteration Level Integration—Integrating UCD Concepts, Roles, and Activities in Agile Development Iteration

This level of integration helps to align UCD concepts, roles, and activities within the development iteration activities for maximum benefit. For example, our framework suggests that UCD tasks exist in addition to the development tasks in every iteration planning. The results of these tasks are presented in the iteration presentation. The level integration resulting effects into agile development are performing iterative design activities, taking measurements, and defining UCD roles.

Iterative Design Activities: In many cases, when UCD techniques are used (if at all), the design of the system is refined according to the users' evaluations mainly during the design phase. In the agile development approach, the design is updated regularly as the product evolves. When combining the UCD approach with agile development, 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 evaluations' ongoing outcomes.
Measures: Taking measurements is a basic activity in software development processes. When combining the agile and UCD approaches, a set of evaluation tools is built and refined during the development process and is used iteratively to complement the process and the product measures.
Roles: Different roles are defined to support software development environments. The agile approach adds roles for better management of the project [17]. Combining the agile and UCD approaches adds UCD roles, such as the design evaluator or the usability expert, to support and carry on UCD activities in the development.

3.3 The Development-Environment Level Integration—Managing and Automating User and Usability Evaluation in IDE

UCD guides integrating user experience into the software development process. One of the challenges of this integration is to automate the management of UCD activities during development. When we analyzed current software design practices, we identified a lack of *UCD management*, which we define as the ability to steer and control the UCD activities within the development environment of the project [6, 7, 12]. Defining evaluation experiments and running them from within the IDE equips the software team with the mechanism to monitor and control a continuous evaluation process, tightly coupled with the development process, thus receiving ongoing user feedback while continuing development.
Experiment Entity: Automating the evaluation methods means that we can add a new kind of an object in the development area of a software project. These objects, known as experiments because of the controlled environment in which they are performed, can be created and executed to provide evaluation data. Furthermore, an experiment's results can be associated with future development tasks as they emerge. We further divide these experiments into three categories: expert-based methods, user-based methods, and system-based methods. Expert-based methods (e.g., heuristic evaluation, cognitive walkthrough) are performed by UCD experts/system experts to evaluate the system usability with respect to standards or guidelines. User-based methods (e.g., question-asking protocol) involve end users performing different tasks on target systems or evaluating the system based on any given criteria. System-based methods (e.g., log file *or* task-environment analysis) use automated evaluation tools to record users' and system behavior and produce analysis results of the recorded data.
Derived development 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 the comparison of these results against the targeted usability criteria. If the results show a failure to achieve the target usability level, then new development tasks can be defined accordingly. Each development task is associated with the relevant data, thus providing its rationale.

Code Traceability: Automating the process of backward and forward traceability among different evolving parts, at the development-environment level provides a better traceability of the refinement carried out in the design to improve the product. Such parts could include code parts, experiments, and derived development tasks.

Developing Evaluation Aspects: Automating UCD activities in the development environment can enable developers to add automatic evaluation hooks to the software under development. For example, an aspect could be created to control the use of a specific button or key that is part of the developing software. These system-based methods that include such measures provide insights about the users' behavior.

4 Automated Tools Support

Our research approach, which involved working with software teams to understand how UCD can be embedded in the software development process, helped us to shape a set of requirements for tooling as well as guidelines and techniques to accompany it [6, 7, 12, 17]. We developed two tools that automate and manage user and usability evaluation at IDE level to provide realization of development environment-level integration.

UEMan: A Tool for UCD Management in Integrated Development Environment

UEMan (**U**ser **E**valuation **Man**ager)**:** [7, 12] is an Eclipse plug-in that supports the automation and management of UCD activities as part of the Eclipse IDE. UEMan includes a Java library of evaluation aspects that enables software developers to integrate automatic measures as part of the developing project. In a nutshell, UEMan evaluation life cycles consist of four phases that occur iteratively.

Evaluation Definition: The evaluation is defined through the following: different types of experiments (such as questionnaires, heuristic evaluations, etc); experiment tasks (which end users/UCD experts need to perform during experiments); role holders who are involved in experiments (e.g., end user, UI expert, evaluator); and other management considerations (such as experiment details, time to execute, etc).

Evaluation Execution: The evaluation is executed by running the experiments either locally at the evaluation site or remotely at the participant's site; thus, tasks in the experiments can be performed by participants and the results can be stored in the system.

Evaluation Analysis: The results are analyzed by software teams as per evaluation experiment and any cross experiments to identify users' and system behavior as well as usability issues.

Feedback to Development: New development tasks for upcoming iterations are derived according to the evaluation analysis for improving the product, whereas connectivity is kept between the evaluation results and the relevant code parts during development.

TaMUlator: A Tool for Managing and Automating Task Model-based Usability Evalaution at IDE level

TaMUlator[1] (**Ta**sk **M**odel-based **U**sability Eva**luator**) is a Java-based tool that provides a set of APIs and interfaces for managing and automating task model-based usability evaluation at the IDE level.

TaMUlator allows the development team to tag tasks and variables of interest at code-level to be used in task models. TaMUlator provides an easy and dynamic way to define different usability scenarios for evaluation. This is achieved by compiling TaMoGolog-based task models that can be aggregated into evaluation experiments, which can then be evaluated at any time by the built-in analyzer using the recorded data of these experiments, or manually evaluated by exporting the recorded data into a CSV format for analysis.

5 Case Studies

We present two case studies in which development teams[2] in academia worked with agile approach for developing their software projects while applying our UCD management approach at IDE level.

In the first case study, six development teams developed an application, named FTSp (Follow the Sun plug-in), to support synchronization between distributed teams that have no synchronous communication between them due to large time zone differences. The development teams (each developed their version of the same application) conducted the evaluation (by other development teams) of the developed application using UEMan by defining and executing the heuristics evaluation experiment and logging-aspect experiment. A total of twelve experiments were conducted to evaluate FTSp. Different teams provided feedback on the contribution of the evaluation process using UEMan. Among other comments, teams mentioned the good collaboration between the team and the participants, the benefit of recognizing the new issues raised that had not been seen before, and the ability of UEMan to automate the results summary, enabling them to identify significant problems and define development tasks accordingly. As part of the evaluation study, we found that software team members engage in UCD management activities in a natural and intuitive manner. They can easily analyze experiments' results for their project and derive significant development tasks accordingly.

In the second case study, six development teams developed their software project through agile development approach while using TaMUlator to evaluate their developed project. The usability evaluation of the product that is being developed was implemented as part of this project. Following are the main practices we used: 1) Iterative design activities that include cycles of development that contain development tasks that were derived from usability evaluation. 2) Role holders in the

[1] TaMUlator demo video:
`http://www.dis.uniroma1.it/~humayoun/tamu/tamulator.html`

[2] The team members were, both times, 4th year CS-major students participating in the 'annual project in software engineering' course of the Computer Science Department at Technion, IIT.

subject of usability evaluation and using TaMUlator. 3) Measurements that were taken by the role holders as part of fulfilling their responsibilities.

In addition to the developing their project, named 'Brain Fitness-Room' aims to develop a system for maintaining and strengthening memory and brain capabilities as well as identifying any decline in these capabilities, the subject of usability evaluation using task models was presented to the teams. In the first development iteration, all teams had to develop a tool with a Java library to enable writing basic task models in a formal way. Based on the teams' work, one tool was selected (TaMUlator) and all teams used this tool during two iterations of evaluation.

In the final retrospective on the course, team members were asked to grade their satisfaction between 1 to 5 (very satisfied) with respect to the project topic, course methodology (agile, time management and early detection of problems, emphasis on testing and usability), tools that were used (Trac and Moodle), and the services in the physical lab they worked in. 32 team members answered and the average grade for the methodology was high (4.09) (for project topic 4.36, tools 3.98/3.28 respectively, and for lab services 2.66). Specifically, regarding the roles that concerned with usability, team members referred to the importance of learning and dealing with usability while developing. Following are some of their comments on this matter: "It is important to get feedback from the users…", "It does not matter how good the product is, [people] will use it only if it is simple and user friendly. A lot of things that seem clear to developers are not clear to the end users," and "The role of being in charge of the evaluation experiment was an important role with which we specified the usage of our system by the user."

6 Conclusion

We have presented a framework that integrates UCD philosophy into agile development practice at three levels. Our framework provides a way to get equal benefits from both approaches, thus enabling the development of high-quality and usable software products. We also presented two automated tools for the realization of the development and that provide environment-level integration and two case studies where they used these tools to evaluate their software projects.

In the future, we intend to apply our framework in medium- to large-scale software projects to properly check its effectiveness. We also intend to work on the presented tools to support better management and automation.

Acknowledgments. Our thanks to Eli Nazarov and Assaf Israel from Technion IIT for their work in developing TaMUlator.

References

1. Agile Alliance. 2001: Manifesto for Agile Software Development. Technical Report by Agile Alliance (2001), `http://www.agilealliance.org/`
2. Balbo, S.: Automatic evaluation of user interface usability: Dream or reality. In: Balbo, S. (ed.) Proceedings of the Queensland Computer-Human Interaction Symposium. Bond University, Queensland (1995)

3. Chamberlain, S., Sharp, H., Maiden, N.A.M.: Towards a Framework for Integrating Agile Development and User-Centred Design. In: Abrahamsson, P., Marchesi, M., Succi, G. (eds.) XP 2006. LNCS, vol. 4044, pp. 143–153. Springer, Heidelberg (2006)
4. Dix, A., Finlay, J.E., Abowd, G.D., Beale, R.: Human Computer Interaction, 3rd edn. Prentice-Hall, Englewood Cliffs (2003)
5. Dubinsky, Y., Hazzan, O.: The construction process of a framework for teaching software development methods. Computer Science Education 15(4), 275–296 (2005)
6. Dubinsky, Y., Catarci, T., Humayoun, S.R., Kimani, S.: Integrating user evaluation into software development environments. In: 2nd DELOS Conference on Digital Libraries, Pisa, Italy (December 5-7, 2007)
7. Dubinsky, Y., Humayoun, S.R., Catarci, T.: Eclipse Plug-in to Manage User Centered Design. In: Workshop on the Interplay between Usability Evaluation and Software Development (I-USED), Pisa, Italy (2008)
8. Fox, D., Sillito, J., Maurer, F.: Agile Methods and User-Centered Design: How These Two Methodologies are Being Successfully Integrated in Industry. In: AGILE 2008, pp. 63–72 (2008)
9. Göransson, B., Gulliksen, J., Boivie, I.: The Usability Design Process - Integrating User-Centered Systems Design in the Software Development Process. Software Process: Improvement and Practice 8, 111–131 (2003)
10. Gulliksen, J., Goransson, B., Boivie, I., Blomkvist, S., Persson, J., Cajander, A.: Key principles for user-centered systems design. Behaviour & Information Technology 22(6), 397–409 (2003)
11. Hazzan, O., Dubinsky, Y.: Agile Software Engineering. In: Undergraduate Topics in Computer Science Series. Springer-Verlag London Ltd, Heidelberg (2008)
12. Humayoun, S.R., Dubinsky, Y., Catarci, T.: UEMan: A tool to manage user evaluation in development environments. In: ICSE, pp. 551–554. IEEE press, Vancouver (2009)
13. Hussain, Z., Milchrahm, H., Shahzad, S., Slany, W., Tscheligi, M., Wolkerstorfer, P.: Integration of Extreme Programming and User-Centered Design: Lessons Learned. In: Abrahamsson, P., Marchesi, M., Maurer, F. (eds.) Agile Processes in Software Engineering and Extreme Programming. LNBIP, vol. 31, Part 3, Part 5, pp. 174–179. Springer, Heidelberg (2009)
14. Ivory, M., Hearst, M.: The State of the Art in Automating Usability Evaluation of User Interfaces. ACM Computing Surveys 33(4), 470–516 (2001)
15. Patton, J.: Hitting the target: adding interaction design to agile software development. In: OOPSLA 2002. ACM, New York (2002)
16. Sy, D.: Adapting Usability Investigations for Agile User-Centered Design. Journal of Usability Studies 2(3), 112–130 (2000)
17. Talby, D., Hazzan, O., Dubinsky, Y., Keren, A.: Agile software testing in a large-scale project. IEE Software, Special Issue on Software Testing, 30–37 (2006)