INTEGRATING UCD WITHIN AN AGILE SOFTWARE DEVELOPMENT PROCESS IN AN EDUCATIONAL SETTING

Olga Ormandjieva, Kristina Pitula*, Cherifa Mansura #*

* Computer Science and Software Engineering Department
Concordia University
Montreal, Quebec, Canada
{olga.ormandjieva, kristina.pitula}@concordia.ca

#TEKsystems
Montreal, Canada
cherifamans @videotron.ca

Abstract – *The Canadian Engineering Accreditation Board has defined 12 attributes that an institution must demonstrate graduates of its engineering program possess. We are in pursuit of the attribute "Design" dealing with the students' ability to select candidate engineering design solutions for development, with three indicators relating to how candidate solutions are selected. Our approach to teaching "Design" is based on "learning outcomes" rather than "teaching inputs". In this paper, we describe the learning outcomes of teaching a newly proposed Integrated User Centered Design (UCD)-Agile Process in the context of a one term project course wherein teams of undergraduate students apply what they have learnt about Agile software development and User Interface (UI) design in the context of a real-world project with actual clients. The Integrated UCD-Agile Process includes upfront design of the UI in parallel with development of the functionalities, UI design specialists for each sprint and usability testing of all UI design decisions.*

Keywords: Canadian Engineering Accreditation Board (CEAB), Design attribute, Agile Software Development Process, User Centered Design, User Interface Design and Evaluation.

1. INTRODUCTION

Quality of education makes a big difference. Students who graduate from accredited programs have had access to better learning opportunities at school and therefore secure better employment opportunities. Accreditation of software courses promotes best practices in education and gives assurance that professionals have a solid educational foundation and can assume leadership roles in industry and life.

In 2010 the Canadian Engineering Accreditation Board (CEAB) defined 12 attributes an institution must demonstrate that its engineering program graduates possess, including Design, Individual and Team Work, Communication Skills and Problem analysis. In this paper we tackle the attribute "Design" that deals with the ability of students to select candidate engineering design solutions for further development.

Over the past few years we have given a one term software engineering project course SOEN 390 at Concordia University wherein teams of undergraduate students apply Agile software development and User Interface (UI) design in the context of real-world projects with actual clients. In the most recent course, we adapted the Scrum methodology followed by the students to more fully integrate User Centred Design (UCD) within the Agile process. As recognized by industry, a Scrum approach has certain deficiencies as to when and how UI design activities are addressed [11]. The development teams face unique challenges when user experience is prominent on a project, such as the issue of content accessibility and the diversity of existing standards that platforms and content are expected to adhere to. Effectively tailoring Agile strategies to meet these challenges requires a scaled approach. Our approach includes the upfront design of the overall UI (in parallel with development of the functional requirements), UI design specialists assigned to each sprint as well as iterative usability testing of all UI design decisions throughout the project.

Comparing the project outcomes to those of previous years, we demonstrate that integrating UCD within the Agile process resulted in a better solution, measured in terms of the UI design evaluation and customer satisfaction.

The remaining part of the paper is organized as follows: Section 2 introduces the required background and surveys the related work. Our approach to student-centred learning in a software engineering project course is explained in section 3. The learning outcomes are discussed in section 4. The conclusions and future work directions are summarized in section 5.

2. BACKGROUND AND RELATED WORK

2.1 Agile Methodologies

The excitement around agile development methodology is undeniable and increasing over time. Most organizations started their agile journey by adopting Scrum and/or Extreme Programming (XP) practices. At the low end of the formality scale, a scrum project management method has no explicit life cycle, no explicit UI definition and no explicit set of UCD deliverables.

These approaches are useful when software projects remain simple and are executed by a co-located team. User stories are the primary vehicle for carrying the customer's requirements through the value stream, from discovery to just-in-time (JIT) analysis, through coding, testing, and deployment. In such smaller, well-contained projects, the lack of UCD deliverables is seldom an issue and if any UI activities are identified, those will be highlighted as tasks (part of a user story implementation) during iteration planning. However, as project complexity grows in size and distribution and as compliance needs and the importance of nonfunctional requirements such as usability increases, a Disciplined Agile Delivery (DAD) approach becomes crucial to project success By providing the framework to integrate varied activities in increments, DAD bridges the gap between requirements, UCD and development better by leveraging a full risk-value cycle, adding agile governance, and specifying the adoption of several complexity-driven practices in the design of UIs and requirements [1].

2.2 User Centred Design (UCD)

User Centred Design (UCD) is an accepted standard for designing interactive software systems that places the focus on end-users to develop systems that meet the users' needs and wants. UCD promotes design as an iterative process in which users are engaged on an on-going basis to drive and refine the design based on feedback and evaluation, and offers a range of established methods and tools for accomplishing this [5], [13]. Nonetheless, producing a "good" design remains difficult, as witnessed by the many poorly designed software products and services in circulation [7]. Among the factors that contribute to good design, understanding end-users, their goals and expectations with respect to the proposed system, validating design decisions with "real" users and designing for the holistic user experience as opposed to localised interaction are key from a UCD perspective [4]. Here the emphasis is on "real" users whose needs are often only partially reflected in client requirements. All involve time and resources to conduct user research and run field studies as well as to analyse, share, integrate and act upon the data collected, with significant effort at the start of a project and more throughout to discover user requirements, validate design decisions and address usability issues as they arise.

In contrast, with Agile methods the focus is on the client. The methodology promotes little upfront design, compressed time frames, design in increments to deliver working software within those time frames and minimal documentation. Given that their respective frames of reference, priorities, dependencies, risks and time frames do not coincide, it is no surprise that integrating UCD within Agile is challenging. A number of recommended practices are emerging from the literature [11] [12]. These include: upfront interaction design in Sprint 0 using UCD tools such as personas, usability scenarios and low fidelity prototypes; basing User Stories on usability scenarios and issues; use of low and high fidelity prototypes to communicate and validate the evolving UI design; and on-going user testing and usability inspections. It is important that UCD practitioners be included as active participants in team meetings and discussions to facilitate communication and establish shared design goals throughout.

2.3 CEAB Attribute "Design"

The Canadian engineering accreditation board (CEAB) mandate tasked each engineering program to assess student outcomes in the form of graduate attributes. The attribute of interest in this paper is "Design". The CEAB "Design" attribute is defined as "an ability to design solutions for complex, open-ended engineering problems and to design systems, components or processes that meet specified needs with appropriate attention to health and safety risks, applicable standards, and economic, environmental, cultural and societal considerations" [3]. A study of the CEAB design skill assessment tools was published recently in [6]. The authors acknowledge the complexity of the design skill assessment as student performance in design is a function of several factors, including design process cognition, discipline-specific knowledge and skills such as team skills, communication skills, project management skills, etc.

The University of Toronto developed concise lists of global objectives and indicators for each attribute. This paper assesses the "Design" attribute in terms of the following three indicators [8] (see also section 3.4):

- *Indicator 1*: Apply formal multi-criteria decision making tools to select candidate engineering design solutions for further development
- *Indicator 2*: Use the results of experiments and analysis to select candidate engineering design solutions for further development
- *Indicator 3*: Consult with domain experts and stakeholders to select candidate engineering design solutions for further development

3. INTEGRATING USER CENTRED DESIGN WITHIN THE AGILE PROCESS

In the past few decades there has been a general trend in higher education towards student-centred learning addressing

the following main objectives: i) acquisition of knowledge, ii) acquisition of skills to extend and improve one's own knowledge, and iii) acquisition of professional problem-solving skills [9]. An example of such a learning approach in software engineering education is the teaching of software engineering project courses where the above-mentioned objectives for higher education are simultaneously addressed.

Our approach to student-centred learning in a software engineering project course is based on "learning outcomes" rather than "teaching inputs". Other educational methods such as lectures and skills training in tutorials are also present, but only to support the student-centred learning.

3.1 Course design

The software Agile process followed in this course is Scrum with an emphasis on team work based on self-direction and collaboration with the clients. Students are grouped in teams consisting of 7-8 team members, with a total of 5-6 teams per class; all teams are given the same set of user requirements to ensure fairness in course evaluation and team competition. The course project is structured in blocks of 6 iterations (sprints), with each block centred on a user stories theme; each sprint is time-boxed to be completed in 2 weeks. The core of each sprint consists of a series of tasks. Team meetings with the clients are held before each development sprint to clarify the user stories, formulate work plan objectives, and communicate the current status of the project. In addition, the agile teams practice short daily collaborative task-based meetings (10-15min). The meetings include individual goal setting for the day (What am I going to do?), strategy selection (How will I do it?) and goal evaluation of the work completed the day before (Did it work?). The time-boxed daily meetings require students to set focused questions for other teammates and respond to inquiries from other team members; the above emphasizes student-centred learning and improves student communication skills. Peer review is undertaken before code is checked in, which increases quality awareness and the result-oriented contribution of individual team members to the group work tasks. At the end of each sprint, the teams have to present a progress report to the other development teams, the instructor, the tutor and the clients. This involves clearly demonstrating and explaining their progress to several different panels. The process helps the students to develop their presentation and communication skills.

3.2 UI design shortcomings in student projects

In theory, clients are the center of the Agile process, thus one of the main objectives of this course was to learn to work cooperatively and collaboratively with the clients. Following the Scrum methodology exposed the problems of addressing UI design activities late or not at all, with a consequent negative impact on quality-in-use of deliverables and client satisfaction. The main reason, not surprisingly, was a lack of understanding of the clients' needs and of the intended use of the application under development. In many cases, the overall UI consisted of an agglomeration of UI components "tastefully" tacked onto the system as they were developed. While individual UI components might adhere to usability principles, taken as a whole, at best the UIs provided only marginal support for users' goals, expectations and behaviour. The lack of a coherent, overall UI structure is a known shortcoming with Agile approaches [11].

3.3 The integrated UCD-Agile process

To address this problem the course curriculum was modified to fully integrate UCD within the Agile process. In this section we first describe the modifications to the process followed by an assessment of the CEAB "Design" attribute in the integrated process.

To a large extent the modified Agile-UCD process followed the recommendations listed in section 2.2. In Sprint 0, along with the Agile deliverables, teams were asked to deliver personas, usability scenarios and a high-level vision of their proposed UI (low fidelity prototype). The purpose of this was to encourage students to think and talk about concrete people using their system in the real world. Additionally, to promote a holistic view, teams were asked to construct Use Case Maps (UCM) to visually represent the complete set of use scenarios. UCM notation is part of an international standard for modelling user requirements in terms of flow of behaviour superimposed over an optional component structure [2]. In this case it was used to specify and analyse UI capabilities and the flow of interaction in the overall UI from a user's perspective.

In the following sprints, low and high fidelity prototypes were used to inspect and user test design options and decisions for the UI increments. Because of resource limitations much of this work was informal. However, to the extent possible and particularly for critical design decisions, students conducted tests with representative users. The final sprint included a UI inspection based on a usability test protocol for course evaluation purposes. The UI evaluation results are summarized in Table 1.

Table 1: UI evaluation results per sprint (2014)

TEAM	SPRINT 1	SPRINT 2	SPRINT 3	SPRINT 4	SPRINT 5
A_2014	88%	80%	100%	75%	95%
B_2014	86.60%	80%	100%	90%	90%
C_2014	94.40%	70%	100%	90%	85%
D_2014	87.50%	60%	85%	70%	70%
E_2014	73.75%	80%	85%	75%	85%
F_2014	86.50%	80%	70%	75%	80%

A comparison between the UI evaluation results in the 2014's Sprint 5 and the 2013 final results (see Table 2) shows

that the integrated UCD-Agile process netted greater UI design quality achievement than did the traditional Agile methodology used in 2013.

Table 2: Final UI evaluation results (2013)

TEAM	FINAL DEMOS
A_2013	80%
B_2013	82.86%
C_2013	73.33%
D_2013	90.71%
E_2013	61%

With regards to UCD documentation, teams maintained a User Interface Requirements (UIR) document in which all UCD related work was captured. In particular, teams were required to document the rationale for all design decisions and changes as well as potential usability issues for future testing

The project had certain special characteristics that gave purpose to applying UCD tools in creative ways. For one, the clients' requirements included (a) designing for a non-technical, text adverse user population; (b) using an alternative to a grid layout for displaying content and (c) producing a minimalist UI (minimal features and steps to complete tasks). The students' lack of knowledge about this population obliged them to seek information from people outside their peer group. The choice of an alternative layout obliged them to research options, test design assumptions and weigh value-effort tradeoffs. The minimalist goal made them think in terms of good UI design for a "minimum viable product", entirely in line with an Agile philosophy and useful in avoiding feature creep. The project also had two clients with sometimes conflicting requirements, a common problem in industry. Here again, students learnt to apply UCD tools to resolve such conflicts with evidence while the documented rationale for design decisions was invaluable in mitigating volatile user requirements.

3.4 CEAB "Design" attributes assessment

In this section, we describe the assessment of the CEAB "Design" attribute in the integrated UCD-Agile process in terms of the three indicators listed in section 2.3 of this paper. The multi-criteria decision making tools to select candidate UI design solutions for further development included: i) low and medium fidelity prototypes of the UI designs for client feedback before sprint development starts, ii) iterative testing of UI design decisions with representative users throughout the project, and iii) user satisfaction forms filled by the clients to track the teams' progress (RE: *Indicator 1*). The results of usability testing and client feedback were used to derive candidate UI design solutions for further development (RE: *Indicator 2*). UI requirements and design were used by the teams early in the development process as criteria for selecting architecture patterns and for guiding the design solutions. For this purpose teams consulted with instructors, tutors and usability experts while gaining problem-solving skills and increasing their learning experience (RE: *Indicator 3*).

3.5 Agile Retrospective Analysis

Agile teams' retrospective analysis sessions were held at the end of the academic term to address the learning outcomes and to emphasize student-centred learning. Students were reminded about the prime directive – "regardless of what is discovered, everyone did the best job they could, given what they knew at the time" and cultural norms of the retrospective that clearly outline the agile team values and working agreements. First, it is about establishing an environment in which the students can safely expose sensitive topics and manage meaningful dialogue. Second, it is about openness in team communication and a common ground for collaboration. Everyone had a "Voice". To support the retrospective, the teams used a known framework among the many available consisting of the following five parts: 1) Start Doing? 2) Stop Doing? 3) Keep Doing? 4) Do More of? 5) Do Less of? [10]. This helped students discuss what things went well, what could have gone better, and how things could be changed to deliver better. At the end, they captured the retrospective results and actions to be taken to improve the development approach for the next project. They were reminded that lessons learned in this session are not lessons forgotten for the next work they need to accomplish.

4. Discussion of the Results

As a positive learning outcome, the development teams learned to use an agile software development process integrated with UCD. This new UCD-Agile process allowed for fast and direct user feedback and continuous integration of the UI design into the developed product through regularly scheduled UI design evaluation and acceptance testing sessions with users and clients. In response to this feedback, the teams developed better design solutions and continually added new value to the product through user-requested feature implementations. The early integration of UCD techniques not only increased the confidence of the students, but also improved considerably the clients' satisfaction with the teams A, B, C, E, and F's deliverables, as shown by the customer satisfaction forms completed by the clients after each sprint (see Table 3). The declining user satisfaction with team D's deliverables was mainly attributed to lower team cohesion.

However, in contrast to the results in 2014, customer satisfaction levels in 2013 (indicated in Table 4) were significantly lower.

Table 3: Client satisfaction results per sprint (2014)

TEAM	SPRINT 1	SPRINT 2	SPRINT 3	SPRINT 4	SPRINT 5
A_2014	75%	70%	80%	85%	100%
B_2014	70%	78%	90%	100%	100%
C_2014	80%	73%	95%	93%	85%
D_2014	80%	65%	75%	78%	65%
E_2014	75%	55%	95%	90%	80%
F_2014	75%	75%	55%	93%	95%

Table 4: Client satisfaction results (2013)

TEAM	FINAL DEMOS
A_2013	80%
B_2013	40%
C_2013	80%
D_2013	40%
E_2013	60%

Hence, the new UCD-Agile process introduced in 2014 resulted in significant customer satisfaction improvement. The main reason is that the quality of the deliverables in 2014 was certainly higher than in previous years due to the risk mitigation of volatile user requirements achieved through the early integration of UCD in the Agile process. Moreover, the regular peer review sessions taught the students that well-structured and well-commented design and code will eliminate the need for lengthy review by the team and/or tutors. The review process follows in the spirit of the Agile methodology, making the entire procedure more efficient and much easier to inherit and improve later.

The introduction of a small portion of the overall course grade dedicated to customer satisfaction (10%) has improved the students' motivation to communicate and collaborate with the clients. Separate grades in the categories of UI design, software architecture and agility of the process showed significant improvement in student use of UCD tools and in finding correct software architecture solutions while improving their agility over the duration of the project.

Over the course of the project, many students realized to what extent the software industry is comprised of numerous evolving technologies not taught at school. This has motivated the teams to start exploring extra languages and components that may not be seen at school. The students were expected to be independent and capable of quickly leaning things on their own, which they appreciated and considered "a lot of fun".

As general feedback on Agile development, the retrospective sessions revealed that not all teams correctly understood the benefits of agile practices or adopted them properly in their course project. These observations were supported by results from an online agility assessment survey [14] completed by all teams, see Table 5. As a learning outcome, the teams realized the disadvantages of not applying agile practices properly and hopefully learned from their mistakes.

Table 5: Teams' agility assessment results (2014)

TEAM	AGILITY ASSESSMENT
A_2014	87.60%
B_2014	84.34%
C_2014	81.45%
D_2014	77.08%
E_2014	70%
F_2014	82.00%

5. Conclusions and Future Work

The goal of this paper was to propose an innovative way of incorporating both theoretical and practical User Interface (UI) design in teaching software engineering Agile project courses that would build solid knowledge and improve the learning experiences for the undergraduate students taking software engineering project courses.

Our work was exploratory, raising questions for further investigation such as: 1) How effective will the UCD-Agile process presented here be in industrial projects? 2) What types of projects are best suited for a UCD-Agile approach?, and 3) How to adapt a UCD-Agile approach to mobile application development? It is questions like these that motivate our research and will be tackled in our future work.

References

[1] S.W. Ambler and M. Lines. *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*. IBM Press, 2012.

[2] D. Amyot and G. Mussbacher. "User requirements notation: the first ten years, the next ten years." *Journal of software* 6, no. 5 (2011): 747-768.

[3] Canadian Engineering Accreditation Board, Engineers Canada. [Online]. Available: http://www.engineerscanada.ca/accreditation

[4] M. Detweiler. "Managing UCD within agile projects." *interactions* 14, no. 3 (2007): 40-42.

[5] DIS, ISO. "9241-210: 2010." *Ergonomics of human system interaction-Part 210: Human-centred design for interactive systems*.

[6] B. Frank. "Development of processes and criteria for CEAB graduate attribute assessment." *Proceedings of the Canadian Engineering Education Association* (2011).

[7] J. Johnson, and A. Henderson. "Usability of interactive systems: It will get worse before it gets better." *Journal of Usability Studies* 7, no. 3 (2012): 88-93.

[8] S. McCahan, G. Allen and L. Romkey. "Development of the graduate attribute quality assurance process at the University of Toronto." *Proceedings of the Canadian Engineering Education Association* (2011).

[9] J. C. Perrenet, P. A. J. Bouhuijs, and J. G. M. M. Smits. "The suitability of problem-based learning for engineering education: theory and practice." *Teaching in higher education* 5, no. 3 (2000): 345-358.

[10] Retrospective Plans [Online]. Available: http://retrospectivewiki.org/

[11] D. Salah, R.F. Paige, and P. Cairns. "A systematic literature review for agile development processes and user centered design integration." In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, p. 5. ACM, 2014.

[12] D. Silva, T. Silva, A. Martin, F. Maurer, and M. S. Silveira. "User-Centered Design and Agile Methods: A Systematic Review." In *AGILE*, pp. 77-86. 2011.

[13] D. Stone, C. Jarrett, M. Woodroffe, and S. Minocha. *User interface design and evaluation*. Morgan Kaufmann, 2005.

[14] Survey on Agility: "Are you agile enough?" [Online]. Available: https://docs.google.com/forms/d/1koe_Pds51MUVm-xGxVcQorh_GCbUBuJssrJXh_O4Efo/viewform?formkey=dC1UMVdyWjRHT3QxdjV2MDRhTnpkOGc6MQ