SindhUniv. Res. Jour. (Sci. Ser.) Vol. 50 (004) 575-582 (2018)

http://doi.org/10.26692/sujo/2018.12.0093

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

Training for Agile Transformation at Universities: A Case Study Analysis

S SHAHZAD, A KEERIO*, S NAZIR**

Department of Computer Science, University of Peshawar, Pakistan

Received 17th December 2017 and Revised 2nd September 2018

Abstract: With the wide spread use of agile methodologies in software development industry proper training of software engineers to cope with the diversity of agile development process has become more crucial than before. One solution for development organizations is to train and prepare their teams for agile acceptance. Although it is the most common scenario for organizations who wish for agile transformation, but this process is slow and lengthy as teams with zero prior agile exposure feel the burden of agile practices and sometimes become defiant. Another solution is to groom perspective software engineers with agile practices in universities and other training institutions. This solution is less expensive, widespread, and long lasting. The following paper analyzes the proceedings and retrospective of a graduate level course of agile software development delivered at a public sector university. The course is based on Agile skill development using Extreme Programming practices in project based training environment. The course was designed and taught with a research goal to establish applicable and realistic guidelines for organizing and delivering an agile methodology training. Such guidelines are of fundamental importance; as agile methodologies require a profound skill development so that perspective software engineers can accept the technological as well as social transformation that is an integral part of an agile development setup.

Keywords: Agile Training, Knowledge Management, Team Development, Emotional Intelligence.

1. INTRODUCTION

Developing professional Software Engineering (SE) and project management skills is a multifaceted task. The students need to learn not only programming skills but they also need to visualize how to actually apply them in professional software development. Along with programming skills software engineers need emotional intelligence (Kosti *et al.*, 2014), knowledge management (Schneider, 2009), and many soft skills like communication and team work to survive in software development industry (Capretz and Ahmed, 2010), (Ahmed, 2012). Agile methodologies inherently focus on all these crucial aspects of software development. Many people have studied Agile knowledge management in class room and many case studies have been presented where students learn together and share their knowledge. The acceptance of Agile methodologies in class room is as challenging as it is in software development industry (Chan and Thong, 2009). The reason is that working with an Agile methodology not only requires to follow a different development process but also demands change in work habits and to learn new skills. The same thing happens when Agile methodologies are taught in class rooms. Here, both parties (students and instructor) must drop traditional pedagogy, and move towards a more open, self-organizing, and knowledge sharing environment. Agile methodologies in class room have proved to be practical pedagogy instrument not only for the instructor but also for the students (Rodriguez, 2015).The students are less dependent on the instructor and more to the knowledge being delivered. It is an ideal tool for making the students learn more in less time.

This paper presents output of an action research carried out while organizing Agile software development course(s) at a university level. The output of the research is presented in the form of a framework of guidelines for using Agile methodologies in class room which emphasizes on imperatives of Agile training at universities.

2. AGILESKILL DEVELOPMENT

Agile software development methodologies have their roots in “Agile Manifesto" (Highsmith and Fowler, 2001), and have become of greatest interest as they provide a flexible way of tackling with the problems in software development while maintaining basic requirements of completeness and timely delivery of software. of all Agile methodologies, Extreme Programming, known as XP (Beck, 1999) has been practiced and researched both by practitioners and academics, and both parties agree on the importance of XP methodology in training for SE skills.

With a wide spread acceptance of Agile methodologies in the software industry, universities are also challenged with providing appropriate training to

Emails: sara@uop.edu.pk, ayaz.keerio@usindh.edu.pk, snshahnzr@gmail.com
*Institute of Mathematics and Computer Science University of Sindh
**Department of Computer Science, University of Swabi, Pakistan

perspective software engineers, as the Agile paradigm has a different approach from traditional development methodologies (Barroca *et al.,* 2018) To have an effective learning outcome from an Agile methodology course, the traditional lecture based pedagogy style also needs to be revolutionized to suit the Agile culture (Sidky, 2007), just as we need to change organizational culture and developer mindset for Agile adoption in industry (Fuchs and Hess, 2018).Also, to create better Agile teams it is necessary to groom students according to their personalities so that they can play an appropriate role in professional life accordingly (Capretz and Ahmed, 2010).

(Dubinsky and Hazzan, 2007) emphasizes that universities must include Agile methodologies in SE training. Žagar (Žagar, *et al.,* 2008) discussed knowledge management and social issues related with SE. These studies have discussed several important things but do not provide a collective framework for an Agile course structure and organization. XP is different from traditional dogmatic SE as it promotes a humanistic approach by following a set of values, principles and practices, that cultivate the social relationships, communication, and emotional intelligence among development teams, obligatory for a productive environment (Henry and LaFrance, 2006).

Research in the field of SE training and that of XP skill development have both now taken a common ground. Hazzan and Dubinsky have used the concepts of XP to exemplify these principles (Hazzan and Dubinsky, 2006), (Missiroli, 2016). Recently Judd et al. (Judd, 2019) have published a study emphasizing the advantages of use of agile for appropriate and efficient skill development for the dynamic and fast growing software industry. Ochodek has defined a way of using Scrum methodology practices for perspective skilled software engineers (Ochodek, 2018).

In general, many approaches have been adopted for strengthening SEskills. One of these which is widely used and researched is university industry collaboration. Macias (Macias, 2004) proposed that small projects based training is not enough to teach practical aspects of software development. A detailed, close to real representation of real-world software development is needed for students to visualize professional software development process. This can be achieved using university- industry collaborations, team based learning, (Van *et al.*,2000), (Rico, 2009) and knowledge management in within teams (Bjørnson and Dingsøyr, 2009).Bjørnson and Dingsøyrhave identified that Agile companies achieve knowledge management at an upper level due to explicitly defined team roles and clear separation of duties (Milenkovic, 2011).

Hence, the concepts of self-organizing controlled teams, writing of program code in pairs and developer switching among pairs, maintaining sustainable development pace, use of product and process metrics in the form of velocities provide a suitable course structure and design in which student evaluation is also pre-designed. Considering this structure, a university level formal SE skills development course is designed and analyzed (Shahzad, 2010), as described in the following sections.

3. METHODOLOGY

The following sections provide details of the SE skill development Agile course and its research outcomes.

3.1 XP coaching using Agile practices

The course was offered to both graduate and undergraduate students at a university in Austria and was attended by about 100 students. The large number of course attendees allowed the instructor to do experimentation with student teams as well as with course structure. The course was allocated one full working day per week which provided the students consecutive eight hours to work for the course. Students were distributed in 10 teams. Whole course was divided into two distinct phases. **Error! Reference source not found.**provides an outline of course organization and structure including the amount of time spent on each topic covered in the class. The time is mentioned as "XP-days", each of which is equivalent to eight hours (Shahzad, 2010).

3.1.1 Teams formation. Before forming teams participants' personalities and their personal likeness for specific roles were identified using a personality analysis questionnaire. The questionnaire included questions regarding general aptitude, social and communication habits, programming experience and knowledge. The teams were then made by equally distributing expertise in all teams, and XP roles of customer, coach, manager, and developer were assigned as per likeness and personality analysis of the students. Extra care was taken to distribute the female students among the teams as they were only the 15% of the total course participants.

3.1.2 Phase 1 – XP Visualization. The objective of this phase was to introduce XP practices in most meaningful and constructive way, and at the same time to provide the students with an opportunity to get ahands-on experience of XP methodology. Knowledge management concepts associated with XP practices were elaborated so that the students get optimal learning benefit from the course. All XP practices taught in XP-Visualization phase were carried out as short exercises using concepts of flipped class room and

active learning (Herreid, 2013) which is essential for faster distribution of knowledge among student. The instructor's own experience with learning XP methodology (Shahzad, 2009, September), (Shahzad, 2009, April) prompted her to place emphasis more on providing training in the XP practices than on scope of the application to be developed in the second phase of the course. The mode of active learning was further enhanced by implementing role playing strategies (Henry and LaFrance, 2006), use of training devices, and use of simulation in SEskill development concepts (Drappa and Ludewig, 2000)in all exercises carried out in this phase. Each exercise was meant to teach particular XP practices.

Table 1. XP Course structure

Phase 1: XP Visualization Phase		**Phase 2: XP realization Phase**	
Topic	Duration	Topic	Duration
Introduction and orientation	1 XP day	Release 1	
XP hour	1 XP day	Iteration 1	2 XP days
Planning Game	1 XP day	Iteration 2	2 XP days
Usability exercise	1 XP day	Release 2	
Test Driven Development exercise	1 XP day	Iteration 1	2 XP days
Mid-term examination	¼ XP day	Project display Preparation	1 XP-day
Mini-project	¾ XP-day	Final exam and project presentations	1 XP-day

Following is a brief description of the learning exercises performed in visualization phase.

XP hour. This full day exercise involved students in XP roles, imposing co-location, story writing exercise, prioritization and estimation, demonstrating pairing concepts, acceptance testing, concept of releases, iterations, and velocity calculation. In XP hour a non-programming but constructive assignment was given to teams of students who completed the assignment in a collaborative development manner.

Planning game. It was a one full day activity involving planning process details, role of on-site customer, feature identification, story template, and estimation. The students were briefed about the concept of planning according to previous iteration's velocity. The aim was to let the students understand all important details of XP planning process. This is important as the experience shows that novice XP teams are feared to get lost in finding an appropriate and fitting planning process for the project as well as for the team (Hussain, *et al.,* 2008), (Shahzad, *et al.,* 2008). The planning game exercise was carefully designed according to students' present knowledge about XP. The story writing guidelines provided them a quick understanding of writing unambiguous stories.

Usability exercise. Concept of designing paper mock-up and introduction to usability testing was provided. The idea was to show the students how usability testing process could be integrated into the XP life cycle.

Test Driven Development (TDD) exercises. TDD workshop was arranged to teach basics of test driven development and the concept behind test-first programming. The idea was to make students understand TDD to make them to understand organized programming. The exercise involved clear introduction unit testing, code sharing, and code integration (Shahzad, 2010).

3.1.3 Phase 2-XP Realization. The teams became more visible when they started working on the project. Before starting with the original project the instructor ran a mock project involving all teams. It demonstrated the way the teams had to work for the rest of the course. The mock project was a small programming assignment to be completed in one day using XP methodology. The actual project was worked on for the rest of realization phase. The following notable activities were performed during this phase.

Standup meeting. For the whole realization phase, each XPday started with a standup meeting conducted by the instructor along with all teams. All team managers provided status report and proceeded to their respective shared workplace, which each teams arranged for themselves, in close vicinity to each other.

Common project definition. All teams worked on same project with some preliminary design requirements, but decision about the major functionality was left up to the customers of the teams.

Visits by company CEO. As the idea was to simulate industry environment, the teams were given the status of small development companies and the instructor designated herself as CEO of the companies. The instructor payed many formal and surprise visits to each team. The focus was on assessing teams' overall performance and members' individual output, and to check if the teams were facing any problems. The instructor followed predesigned protocols to make the visits productive and beneficial.

Changing requirements. The concept and effect of changing requirements was introduced by the instructor by changing basic important features of the desired application after some time for which the teams had to fix their planning accordingly.

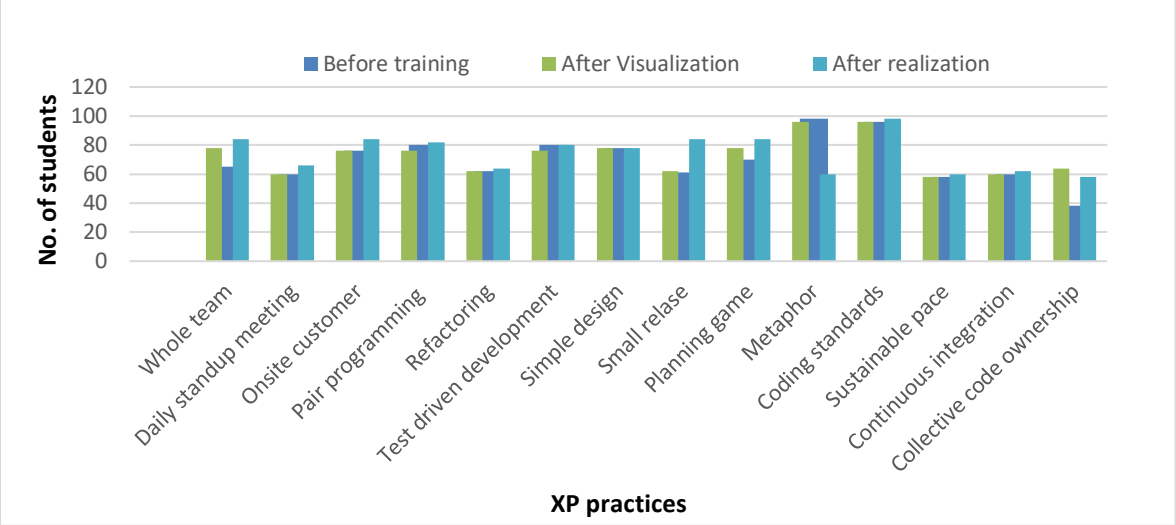

Fig. 1: Learning curve of XP practices

Trade show (projects presentation). At the end of the realization phase, a presentation day was arranged when all teams formally presented their project. Managers, customers, developers, and UI designers all came forward and described their role. This provided the students an experience of formal business presentations.

Seminars for managers and customers. To enhance the knowledge about management and customer's responsibilities, the instructor invited a business person working in a real software development company to share his experience of working as a customer in an XP team. Similarly, a seminar was conducted to discuss software project managers' duties. These seminars were highly appreciated by the students.

Agile acceptance surveys. In the context of the research interests of the author, three formal surveys were conducted during the course. The questions of the surveys were focused on gathering information regarding the applicability of XP methodology as an SE course and also the acceptance of the XP practices by the students. The surveys were scheduled before starting the course, after visualization phase and after realization phase. A brief analysis of the surveys is provided in **(Fig. 1).** Along with the surveys informal feedback from students was also collected through planned and unplanned discussion sessions, planning and retrospective meetings during the whole semester, and also from the course's wiki forum which contributed to promote the project management and industry simulation concepts of the course (Shahzad, 2010).

3.1.4 Analysis of Agile acceptance surveys. From **(Fig. 1).** it can be seen that generally the students learning improved through the whole course. Forsome practices learning trend was more visible and encouraging, for example, the concepts of "whole team", "onsite customer", and "small releases" was well understood and accepted by the students after the realization phase. The students already knew some practices, for example, "simple design", "test driven development". There were also practices which students found hard to understand, for example, "metaphor", even after the XP realization phase."Collective code ownership" was a practice that made clear impact after students worked on the project in realization phase. It showed them the actual spirit of team work and sharing. This course review showed that the students learned to use the practices of XP more in realization phase than after the visualization phase, which was the intention of

dividing course into separate phases of theory and practice.

4. Framework of Agile course and course management

The research goal of this course was to define a framework for structure and organization of an Agile software development methodology course to analyze the learning issues of XP practices and Agile knowledge management. The desired objective was to provide students a meaningful experience with learning an Agile software development methodology which becomes useful and helpful for them professional life. The idea was to develop an innovative way of XP training so that the students not only learn XP as a software development methodology but can also visualize the development process as an organized activity to achieve a goal. Another important aspect was to expose and emphasize the knowledge management aspects of individual XP practices and to demonstrate how formal as well as informal knowledge management helps in the learning process.

The course helped to build a culture of student involvement in learning activities and a knowledge sharing environment. It also highlighted the importance of communication among students by integrating social structure of XP practices and role playing activities with pedagogical techniques of communication.

(Table 2) presents the proposed framework which defines the impediments for the organization and structure of an Agile development methodology course.

Table 2.Framework of Agile course management (Shahzad, 2010)

Course Organization	Clearly distribute course content and time into theory and practice phases. It makes the course manageable, adaptable, and well structured.
	Instead of lecture based pedagogy use innovative training aids (board exercises, audio video presentations, project based games, workshops) to make the class more interactive, this optimizes learning outcome.
	Design lectures as short, to-the-point, and time boxed workshop style exercises, which makes it easy to teach fundamentals of Agile.
	Short on-spot programing exercises for pair programming and test driven development practices provide better training to the students.
	Follow a pre-set schedule for theory and practical exercises. Although it is against the spirit of agility but too much agility in the class room will result in chaotic situation, especially during workshops, and the goals of the workshops will not be met.
	Make sure that each student participates in workshops and exercises.
	Assess student performance after workshops/exercises to make the students realize the significance of the exercise.
Team distribution	Make small teams, of about five to eight students.
	Make teams in such a way that expertise and knowledge is fairly distributed among all teams.
	Make an even distribution of male and female students. Female students are usually good in general management of teamwork.
	Instead of directly starting with fixed teams let the students know each other in the whole class by making random teamsin the beginning. Working in fixed teams from the very beginningrestricts the knowledge distribution and is also againstthe general philosophy of agility.
Roles	Introduce roles in the teams. Roles defined by XP are ideal for programming as well as management related skill development.
	It is better to take students' consent in assigning roles. Impose the role if one is not available in the team. Employ some general technique for personality evaluation before assigning roles.
	Assign roles before making teams.
Project Organization	Define fixed time slots for iterations and releases. Make sure that each team follows that schedule.
	Supervise the work of each team during project time and make sure all the practices are being properly followed. If any practice(s) is not properly followed provide more training in this regard.
	Clearly specify activities for each role. For example, in planning game explicitly define the roles of managers, customers, and developers.
	Make sure that all team members have tasks to do all the time. For example, when developers are busy in coding, arrange special exercises for customers and managers.
	Define a schedule of daily routines during project and make sure that all teams follow it. Specify the schedule as minimum requirement and leave room for additional activities so that the teams can also learn to organize themselves.
	Define protocols for the duties of managers, customers, coaches, and developers during project.
	Closely supervise teams while they are working. The instructor must visit each team during project. A protocol defining what to look for during visits to the teams will be helpful in comparing and evaluating activities of different teams.
Communication and Collaboration	Provide a collaboration platform for all students and teams so that they can communicate as well as share knowledge easily and efficiently. For example, setting up the course wiki and allowing the students to manage personal as well as team portals. It provides a way to supervise team activities.
	In case of some social or communication related problem among team members ask the manager to work as moderator and resolve small issues occurring in team work.
	It is also ideal to have student assistants for supervising and guiding the teams during project. Define a protocol of duties for the student assistants.
	Simulating the corporate industry style communication link among the course organizer, managers and customers will also provide a good learning and training experience for managers and customers.

5. CONCLUSION

The main purpose of this course was to disseminate the importance and practice of Agile approach for SEskills development. The usage of Agile methodologies is an unnerving challenge. Attempt was made to gradually impart the principles and philosophies of Agile methods to the students in the duration of 15 weeks. The course proved successful in exposing various aspects of actual software development environment in a simulated way. The students learned how to organize themselves for a productive team work using the technology and practices which are a norm in industry. The course provided an opportunity to students to analyze and boost their expertise and skills related to technology as well as soft skills and emotional intelligence. The recommendations from that formal course organization and implementation provided an opportunity to set guidelines for providing such a productive and conclusive learning experience to students. Finally, it is conceived that a formal course design and organization is necessary to teach Agile methodologies if the aim is to teach Agile principles in full spirit.

REFERENCES:

Ahmed, F. C. (2012). "Evaluating the demand for soft skills in software development. *IT Professional, 14*(1), 44-49.

Assassa, G. (2006). Extreme programming: A case study in SE courses. *Proceedings of the 1st National Information Technology Symposium, NITS, Riyadh, Saudi Arabia*,. 233-240

Barroca, L., H. Sharp, D. Salah, K. Taylor, and P. Gregory, (2018).Bridging the gap between research and agile practice: an evolutionary model. *International Journal of System Assurance Engineering and Management* 9(2), 323-334.

Beck, K. (1999). *Extreme Programming Explained: Embrace Change (1st Edition).* Addison-Wesley Professional.

Bjørnson, F. O., and T. Dingsøyr, (2009). A survey of perceptions on knowledge management schools in Agile and traditional software development environments. *XP*, 94-103.

Capretz, L. F., and F. Ahmed, (2010). Making sense of software development and personality types. *IT professional, 12*, 6-13.

Chan, F. K., and J. Y. Thong, (2009). Acceptance of Agile methodologies: A critical review and conceptual framework. *Decision Support Systems, 46*, 803-814.

Drappa, A., and J. Ludewig, (2000). Simulation in SE training. *ICSE '00: Proceedings of the 22nd international conference on SE*. 199-208.

Dubinsky, Y., O. Hazzan, (2007). Why SE programs should teach Agile software development. *SIGSOFT Softw. Eng. Notes, 32*(2), 1-3.

Fuchs, C., and T. Hess, (2018). Becoming Agile in the Digital Transformation: The Process of a Large-Scale Agile Transformation. Proceedings of the 39th International Conference on Information Systems (ICIS 2018), San Francisco, USA.

Hazzan, O., and Y. Dubinsky, (2006). Teaching framework for software development methods. *ICSE '06: Proceedings of the 28th international conference on SE*, 703-706.

Henry, T. R., and J. LaFrance, (2006). Integrating role-play into SE courses. *J. Comput. Small Coll., 22*, 32-38.

Herreid, C. and N. A. Schiller, (2013). Case Studies and the Flipped Classroom. Journal of College Science Teaching, *42*(5), 62-66. 84

Highsmith, J., and M. Fowler, (2001). The Agile Manifesto. *Software Development Magazine, 9*, 29-30.

Hussain, Z., M. Lechner, H. Milchrahm, S. Shahzad, W. Slany, M. Umgeher, and T. Vlk, (2008). Optimizing Extreme Programming. *ICCCE 2008: Proceedings of the International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia*1052-1056.

Judd, M. M., and H. C. Blair, (2019). Leveraging Agile Methodology to Transform a University Learning and Teaching Unit. In *Agile and Lean Concepts for Teaching and Learning*, 171-185.

Kosti, M. V., R. Feldt, and L. Angelis, (2014). Personality, emotional intelligence and work preferences in software engineering: An empirical study. *Information and Software Technology*, 56(8), 973-990.

Macias, F. (2004). *Empirical Assessment of Extreme Programming.* Ph.D. dissertation, University of Sheffield.

Milenkovic:, V. D. (2011, May). Teaching Agile Software Development: A Case Study. *IEEE Transactions on Education, 54*(2).

Missiroli, M. R. (2016). Learning Agile software development in high school: an investigation. *Proceedings of the 38th International Conference on SE Companion*, 293-302.

Ochodek, M. (2018). A scrum-centric framework for organizing software engineering academic courses. In Towards a Synergistic Combination of Research and Practice in Software Engineering, 207-220.

Rico, D. F. (2009). Use of Agile methods in SE education.*Agile Conference*, 174-179.

Rodriguez, G. S. (2015). Virtual scrum: A teaching aid to introduce undergraduate SE students to Scrum. *Computer Applications in Engineering Education*, 147-156.

Schneider, K. (2009). *Experience and knowledge management in SE*. Springer Science and Business Media.

Shahzad, S. (2009). Knowledge Management Issues in Teaching Extreme Programming. *9th International Conference on Knowledge Management and Knowledge Technologies, I-Know and I-Semantics,* 278-288.

Shahzad, S. (2009). Learning from Experience: The Analysis of an Extreme Programming Process. *Sixth International Conference on Information Technology: New Generations,* 1405-1410.

Shahzad, S., Z. Hussain, M. Lechner, and W. Slany, (2008). Inside View of an Extreme Process. *XP*, 226-227.

Shahzad, S. (2010). Analyzing the Extreme ProgrammingPractices and Knowledge Managementin Software Engineering Education - Shrinking the Gap between Industryand Academia.(*Unpublished doctoral dissertation). Technical University of Graz, Austria*

Sidky, A. S. (2007). A structured approach to adopting Agile practices: The Agile adoption framework. Diss. Virginia Tech.

Van Solingen, R., E. Berghout, R. Kusters, and J. Trienekens, (2000). From process improvement to people improvement: enabling learning in software development. *Information and Software Technology, 42*, 965-971.