ACM SIGSOFT Software Engineering Notes Homepage archive
Volume 32 Issue 2, March 2007
Pages 1-3

Why Software Engineering Programs Should Teach Agile Software Development

Orit Hazzan
Department of Edu. in Technology & Science
Technion – Israel Institute of Technology
oritha@tx.technion.ac.il

Yael Dubinsky
Department of Computer Science
Technion – Israel Institute of Technology
yael@cs.technion.ac.il

Abstract

In this paper we propose ten reasons why it is important, suitable and timely to introduce agile software development into software engineering programs in the academia. These reasons address technical, social and cognitive issues.

Keywords: software engineering, agile software development, teaching agile software development, software engineering education.

Introduction

In this paper we propose ten reasons why it is important, suitable and timely to introduce agile software development into software engineering programs. Our contribution is based on our comprehensive teaching and research experience with respect to agile software development, both in the industry and in the academia, during the last five years.

Our paper is aimed at two main communities: software engineering practitioners and software engineering educators. Since the adoption by the industry of the agile approach is becoming mainstream and is taking place in practice, software engineering practitioners can use the arguments presented here to explain the industry's interests and needs to people in academia; software engineering educators can use the arguments presented in this paper to improve their understanding of why agile software development should be taught at university level.

Why teach agile software development?

This section presents ten arguments, presented in short in Table 1, why software engineering programs should include agile software development methods (hereinafter referred to as AGILE). The order of the presented arguments does not necessarily reflect their importance.

1. AGILE was evolved and is applied in the industry
2. AGILE educates for teamwork
3. AGILE deals with human aspects
4. AGILE encourages diversity
5. AGILE supports learning processes
6. AGILE improves habits of mind
7. AGILE emphasizes management skills
8. AGILE enhances ethical norms
9. AGILE highlights a comprehensive image of software engineering
10. AGILE provides a single teachable framework

Table 1. Ten reasons why software engineering programs should teach agile software development

1. AGILE was evolved and is applied in the industry: AGILE was evolved by practitioners working in the software industry (cf. the case of Extreme Programming, developed by Kent Beck), and clearly, it has become mainstream. An illustration which reflects the massive attention AGILE is receiving, is the vast attendance at the Agile conferences (e.g., more than 1,000 participants attended the Agile 2006 conference that took place in Minneapolis in the Summer of 2006). As AGILE is becoming prevalent in the industry, its teaching in the academia is just natural.

2. AGILE educates for teamwork: According to the Software Engineering 2004 Curriculum[1], software engineering students should acquire skills, beyond the technical and scientific ones, such as teamwork-related ones. Since agile software development is based on teamwork [2], it is only natural to integrate teamwork-oriented skills, such as retrospective processes and communication skills, in the teaching process of agile software development. Further, since it is most natural to teach agile software development in a teamwork-oriented environment, there is no need to introduce the topic of teamwork artificially; rather, a teamwork-based teaching environment can and, in fact is recommended, to be used in the teaching of agile software development for the introduction of teamwork-related topics.

3. AGILE deals with human aspects: DeMarco and Lister [3], Tomayko and Hazzan [12], as well as others, acknowledge the importance attributed to human (i.e., cognitive and social) aspects of software engineering. Accordingly, human-related topics involved in software development processes should receive more attention in software engineering education. Since two of the agile manifesto ideas (http://www.agilemanifesto.org/, see Table 2), namely *Individuals and interactions over processes and tools* and *Customer collaboration over contract negotiation,* specifically address people (developers, customers and other stakeholders), it seems that agile software development may be an appropriate ambassador for delivering the importance of human aspects of software engineering as well.

4. AGILE encourages diversity: Diversity can be expressed in terms of culture, perspectives, gender, minorities, life styles, nationalities and other factors. Regardless of the shape the diversity takes, studies tell us that diversity benefits societies that welcome

[1] Software Engineering 2004 - Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, A Volume of the Computing Curricula Series, developed by The Joint Task Force on Computing Curricula IEEE Computer Society and the Association for Computing Machinery, http://sites.computer.org/ccse/SE2004Volume.pdf.

it in general (see for example [6]), and software development teams in particular [1]. In addition, in the context of the software industry, since more and more companies are going global, diversity is becoming an integral characteristic of software development teams [5]. If software organizations are indeed becoming more diverse, then we should let our students experience diversity and its benefits, manage diversity and explore how it can be achieved. Agile software development might be an appropriate teaching framework for achieving these educational targets, since diversity is derived from agile principles (e.g., customer collaboration) and is expressed by agile practices (such as informative workspace, pair programming and planning game). Further, the establishment of diverse teams (see Reason no. 2) in educational programs in general and in the teaching of agile software development in particular might be a good opportunity to highlight this message.

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Table 2. Manifesto for Agile Software Development

5. AGILE supports learning processes: It is well known that software development is a complex challenge from the cognitive perspective as well. For example, both customers and developers should improve their understanding with respect to the developed product throughout the entire software development process. In other words, software development processes are, in fact, learning processes that should be supported by the development environment. As it is explained in what follows, AGILE does support learning processes. One mechanism that reflects this is the concept of small releases, which means that the developed product is delivered in short iterations, at the end of each such iteration, the customer and the developers have an opportunity to rethink what has been developed, reflect back on what has been accomplished, and proceed based on the lessons learned. Another practice of agile software development that supports learning processes is refactoring ([8]). Thus, AGILE might reduce part of the cognitive complexity inherent in software development processes by making the development environment more comprehensible to the developers, customers and other stakeholders. Clearly, software engineering students should be exposed to such development environments.

6. AGILE improves habits of mind: A lot has been said about the mental skills that software engineers, as well as software engineering students, should possess in order to produce quality software. Among them we can mention reflection ([7], [9], [11]), abstraction skills ([10]) and program comprehension ([13]). Such skills can be fostered very naturally by agile software development in general, and in particular by activities such as stand-up meetings, pair programming, and small releases, that are an integral part of any agile software development environments.

7. AGILE emphasizes management skills: Many objections expressed against AGILE state that agility removes the responsibility from the team leader. In practice, however, in agile development environments, leadership is distributed among all team members, or, in other words, all team members share the responsibility for the developed product. Accordingly, if AGILE is taught at university level, students may also gain some software management skills.

8. AGILE enhances ethical norms: The Software Engineering Code of Ethics and Professional Practice (http://www.acm.org/serving/se/code.htm), formulated by an ACM/IEEE-CS Joint Task Force, outlines how software developers should adhere to ethical behavior. The eight principles of the Code of Ethics are presented in Table 3. As the following illustration demonstrates, agility supports ethical behavior. The second principle of the Code states that "software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest." The correspondence of this principle to the Agile Manifesto's concept of "Customer collaboration over contract negotiation" (see Table 2) is transparent: they both give priority to the customers' interest. Thus, teaching agility becomes a perfect opportunity to address also the code of ethics of software engineering.

1 PUBLIC - Software engineers shall act consistently with the public interest.

2 CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.

3 PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

4 JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.

5 MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

6 PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

7 COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.

8 SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Table 3. The Principles of the Software Engineering Code of Ethics and Professional Practice

9. AGILE highlights a comprehensive image of software engineering: The fact that the arguments presented so far relate to different fields (such as cognition and management) reflects the fact that agile software development paints a comprehensive picture of software development processes, as well as highlights connections between the elements and players participated in a typical software development process. Thus, AGILE might serve also as an opportunity to expose software engineering students to this picture.

10. AGILE provides a single teachable framework: Software engineering curricula are usually composed of different topics, sometimes taught as isolated topics. Since in agile software development the team concept is accepted and all developers are involved in design, coding and testing activities, an agile environment provides, in fact, a single all-inclusive teaching framework for software engineering. Clearly, specific additional courses should also be taken in order to deepen the students' knowledge about the main topics of software development.

Epilogue

Convinced readers may ask themselves at this stage how the actual teaching of agile software development should be carried out. Since the purpose of this paper is to present our perspective about why it is important to introduce agile software development into software engineering programs in the academia, we do not elaborate here about the actual teaching of AGILE. We will just mention that in [4] we introduce a teaching framework that encompasses all the ideas mentioned above. Within this framework, which can be implemented within one semester period, one version of a software product is developed in three iterations. Interested readers are welcome also to visit at our website[2] and to contact us for further information and/or collaboration.

References

[1] Beck, K. with Andres, C., 2005. *Extreme Programming Explained*, Addison-Wesley.

[2] Cockburn, A., 2001. *Agile Software Development*, Addison-Wesley.

[3] DeMarco, T. and Lister, T., 1999. *Peopleware: Productive Projects and Teams*, Dorset House Publishing Company.

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

[5] Ebert, C. and DeNeve, P. 2001. Surviving global software development, *IEEE Software* **18**(2), pp. 62-69.

[6] Florida, R., 2002. *The Rise of the Creative Class*: *And How It's Transforming Work, Leisure, Community and Everyday Life*, Basic Books.

[7] Hazzan, O., 2002. The reflective practitioner perspective in software engineering education, *The Journal of Systems and Software* **63**(3), pp. 161-171.

[8] Hazzan, O. and Dubinsky, Y., 2003. Bridging cognitive and social chasms in software development using Extreme Programming, *Proceedings of the Fourth International Conference on eXtreme Programming and Agile Processes in Software Engineering*, Genova, Italy, 47-53.

[9] Kerth, N. L. 2001. *Project Retrospective*, Dorest House Publication.

[10] Kramer, J. in press. Abstraction – the key to Computing? *Communications of the ACM.*

[11] Schön, D. A. 1987. *Educating the Reflective Practitioner: Towards a New Design for Teaching and Learning in The Profession*, San Francisco: Jossey-Bass.

[12] Tomayko, J. and Hazzan, O., 2004. *Human Aspects of Software Engineering*, Charles River Media.

[13] Vans, A. M., von Mayrhauser, A. and Somlo, G., 1999. Program understanding behavior during corrective maintenance of large-scale software, *Int. Journal Human-Computer* Studies 51, pp. 31-70.

[2] Information about our work can be found in the following URL: http://edu.technion.ac.il/Courses/cs_methods/eXtremeProgramming/XP_Technion.htm