Capable Leader and Skilled and Motivated Team Practices to Introduce eXtreme Programming

Lech Madeyski[1] and Wojciech Biela[2]

[1] Institute of Applied Informatics, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50370 Wroclaw, POLAND
Lech.Madeyski@pwr.wroc.pl, http://madeyski.e-informatyka.pl/
[2] ExOrigo Sp. z o.o., Krucza 50, 00025 Warsaw, POLAND
Wojciech.Biela@exorigo.pl, http://www.biela.pl

Abstract. Applying changes to software engineering processes in organisations usually raises many problems of varying nature. Basing on a real-world 2-year project and a simultaneous process change initiative in Poland the authors studied those problems, their context, and the outcome. The reflection was a need for a set of principles and practices to help introduce eXtreme Programming (XP). In the paper the authors extend their preliminary set, consisting of the Empirical Evidence principle, exemplified using DICE®, and the practice of the Joint Engagement of management and the developers. This preliminary collection is being supplemented with the Capable Leader, as well as the Skilled and Motivated Team practices based on the DICE® framework as well.

Key words: Extreme programming, Agile adoption, Process change, Software process improvement, DICE® framework.

1 Background

The paper is based on a real-world software project and the attempt to bring agile practices to that project and organisation. The organisation is a medium-sized company that operates in several distant locations in Poland. The study relates to a 2-year project developed by 3 programmers (the whole team consisted of 8 programmers). The goal of that project was a B2C web platform for a trust fund agent.

One of the authors joined the team, after a year from the project start, to resolve various issues that arose in the development environment. At that time, his only knowledge and experience concerning agile practices came from the e-Informatyka project led by the co-author, their long discussions and of-the-books knowledge. The problems were addressed using a collection of agile techniques.

Test-Driven Development (TDD) was something new and was a complete success. The recognised overhead in writing tests was compensated by the decrease in the bug rate (which is in line with [1]) and the ability to refactor the system quickly. It was not without resistance and a lot of coaching was necessary, but eventually developers found TDD to be very effective and rewarding.

Refactoring was used in the past, but not that explicitly and often. Without doing regular refactorings (while having unit tests in place), many change requests from the client would have met very strong resistance from the team and would have caused a lot more pain.

Pair Programming (PP) was introduced at some point and it really helped the team to share knowledge and halve the time of introducing a new person to the project. Developers realized that PP also helped produce better code in terms of design quality and the number of bugs discovered later. However, not all of the empirical studies [2–8] support the positive impact of PP on software quality, as was observed by the team. Substantial problems arose when the consequences of pairing - higher costs - were exposed to the client. On the basis of several empirical studies [9–13, 6], one may conclude that pair programming effort overhead is probably somewhere between 7% and 84%, whilst the team's observation was close to a 50% overhead.

In-process design sessions required a lot of coaching and basic programming recommendations [14] were introduced. More advanced principles like Separation of Concerns and Dependency Injection were introduced as well. There was some resistance in this matter as with TDD. But the return on investment in this case was usually very high, so discussions on this were swift.

Problem Decomposition was, alongside TDD, the most successful technique brought to the team. Divide the problems into pieces that you can grab, solve the problems (estimate, plan, design or code, whatever the case), and get back to the whole. Whether used at the release planning level or at the implementation detail level, it performed brilliantly.

Continuous Integration and task automation was an obvious benefit, the team saved many hours of dull deployment work. Also many man-made mistakes were omitted due to task automation.

Darts were something completely new, but this great incentive ultimately glued the team together. People started talking to each other. They suddenly had another motivation to complete their tasks (they could have a game). Additionally it provided yet another reason, other than the biological one mentioned by Beck [15], to take your eyes off the computer screen, get up, and clear your head. Eventually, the management accepted it (and had a game themselves). In the authors' opinion, such group toys are a must have for any development team.

Communication was and still is an issue because the team is remote. A wiki was set up, which helped a lot. Direct communication with the customer was encouraged. A conference area was set up, with Skype installed. Much of the outdated documentation was disregarded, instead user stories were recorded for further discussions. This also met resistance, as the customer was in the habit of doing things the traditional way.

The preliminary results, emphasising the need for a concrete set of principles and practices that would complement the main body of eXtreme Programming (XP) and support the fragile process of introducing XP practices, have been presented and discussed within the agile community [16]. These results consisted of an agile principle (*Empirical Evidence*) and practice (*Joint Engagement*) pro-

posal to aid the process change. The *Empirical Evidence* principle recommends to ground on empirical evidence when introducing changes. One of the widely accepted sources of empirical evidence concerning introducing changes is the DICE® framework [17], created by The Boston Consulting Group. However, other sources of empirical evidence are welcome as well. The *Joint Engagement* practice is guided by the *Empirical Evidence*, as well as the *Accepted Responsibility* principle [18]. Following the *Joint Engagement* practice we begin the change process at various structural levels of an organisation [16]. This preliminary set of principles and practices will be further extended in the next section.

2 Keep the DICE® Rolling

The DICE® framework is a simple empirical evidence-based formula, based on 225 change initiatives study, for calculating how well an organisation is or will be implementing its change initiatives [17]. The DICE® framework comprises a set of simple questions that help score projects on each of the five factors: project duration (D), team's integrity (I), commitment of managers (C1) as well as the team (C2), and additional effort (E) required by the change process. Each factor is on a scale from 1 to 4. The lower the score, the better. Thus, a score of 1 suggests that the factor is highly likely to contribute to the program's success, and a score of 4 means that it is highly unlikely to contribute to the success [17]. In DICE®, a project with an overall score between 7 and 14 is considered a *Win*, between 14 and 17 is a *Worry* and between 17 and 28 is a *Woe*. The DICE® formula is D+2∗I+2∗C1+C2+E.

The authors used the DICE® and its C1 and C2 factors previously when proposing the *Empirical Evidence* principle and *Joint Engagement* practice duo [16]. The project team's performance integrity factor (I) concerns the ability to complete the process change initiative on time and depends on the team members' skills. According to Sirkin et al. [17]:

> If the project team is led by a highly capable leader who is respected by peers, if the members have the skills and motivation to complete the project in the stipulated time frame, and if the company has assigned at least 50% of the team members' time to the project, you can give the project 1 point. If the team is lacking on all those dimensions, you should award the project 4 points. If the team's capabilities are somewhere in between, assign the project 2 or 3 points.

Building upon the *Empirical Evidence* principle and the DICE® framework the authors recognise that the project team's performance integrity factor (I) may lead to two new practices: the *Capable Leader* and the *Skilled and Motivated Team* practices. Following those practices one improves their DICE® integrity (I) factor and thus increases the likelihood of success. These practices are clearly in line with the *Improvement* principle [6] from XP. Furthermore, other XP principles like *Diversity*, *Flow*, *Quality*, *Accepted Responsibility* [18] are closely related to the ideas behind those practices.

2.1 Capable Leader

This practice proposal turns the attention to the role of the team leader. It may concern both the project's development team leader responsible for implementing the development practices in a project, as well as the leader for the organisational change process which happens throughout the organisation willing to adopt agile practices.

In the adoption of agile practices the role of the team leader is very significant. New ideas and enthusiasm often die on its own. Things get back to where they were before and the chance for a process change is inadvertently lost, as the same organisation will probably not want to try it twice. There needs to be a person who will advocate for the change, explain and remove obstacles. The project team should be led by a highly capable and motivated leader who is respected by his peers. The team leader should serve the team, never the opposite. On a daily basis he is not a manager, but like the Scrum Master in Scrum [20], rather a normal team member, who has to put on different hats according to the current needs of the team. One time he needs a coach's hat, afterwards he takes out a developer's hat, next minute a manager's hat, then again a team catalyst's hat or maybe an agile evangelist's hat. His ultimate role is to help the team improve, not force them to improve, but rather enable them to do it on a daily basis. But be aware that the team leader cannot make the team dependent on his person, among other problems that would simply cause the Truck Factor to go down dramatically. Contrary, if the team lead should leave the team for a week, nothing dramatic should happen.

Both the team and the team leader need specific resources to be assigned for the change process. For example if they do not have enough time and opportunities to roll out the changes then no amount of wisdom nor tooling will help. Other than that there is no fixed set of resources they need, the whole team together has to identify their requirements.

2.2 Skilled and Motivated Team

This practice in turn focuses on the need to grow a team of motivated professionals and care for them. The major problem is the difference between a team and a work-group and how to help a work-group become a team. Again, this may concern the team of developers implementing a product or more broadly the entire team of the individuals responsible for the change process.

It is quite obvious that skilled professionals are more effective than the weaker ones, but the point is to explicitly aim for having the best people on board. One should not stop there, because very talented individuals often don't work well together, this is why this practice focuses on teams rather than individuals. The people forming such a team need to be motivated to effectively work towards common goals. That is what differentiates them from a work-group of clever masterminds working alone in the same room (often barely talking to each other). Agile processes adoption is no exception from that rule. It is good to form the

team in such a way that it has a critical mass of agile believers and practitioners. Then one needs to help them work together making heavy usage of team retrospectives [19]. Act, inspect and adapt. If it is not possible to change the crew, then coach them extensively, and make even more use of retrospectives.

Following the proposed practices helps to assure the highest capability, skills and motivation of both the team leader and the team members. It means that we do our best to have the project team led by a competent and motivated individual who is respected by his peers. Moreover, it is best when the team members have the skills and motivation to complete the project in the accepted time frame [17]. To achieve these aims the organisation has to assign a reasonable part of the team members' time to the change process. Among other activities that would have an impact on the team's integrity factor are e.g. assuring an appropriate degree of financial support, emphasising the understanding of the potential contribution of the change process to the situation of the team, or a particular team member. Individuals at the management and at the developer level should be educated and involved in the process (*Joint Engagement* practice [16]). They have to willingly accept their diverse responsibilities in the change process (*Accepted Responsibility* principle [18]). The current form of the proposed collection of principles and practices to introduce XP, against a background of the main body of XP, is presented in the Figure 1.

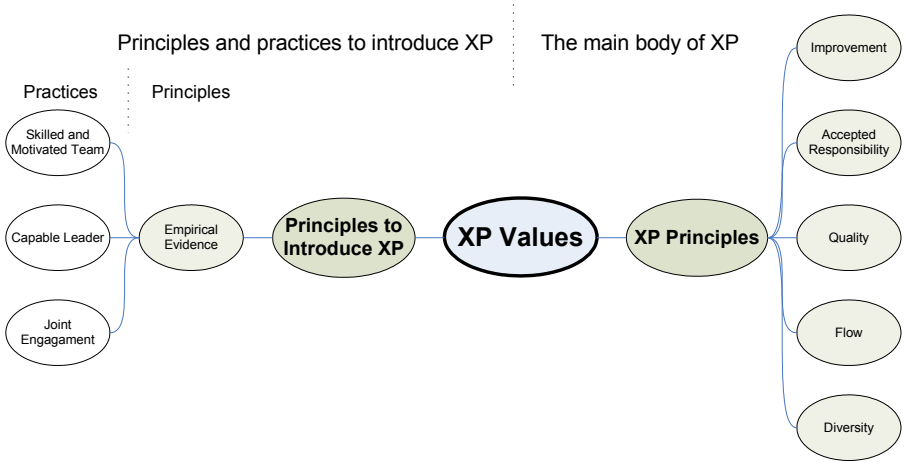

Fig. 1. The set of principles and practices to introduce XP against a background of the main body of XP

3 Conclusions

Introducing agile practices is a challenge for every organisation. In the search for methods that would ease the adoption process, the authors began to identify

principles and practices to introduce XP [16]. In the paper they identified further two potential practices based on the Empirical Evidence principle: the *Capable Leader* and the *Skilled and Motivated Team* practices. Such practices might not be very surprising for some. However, as the history of XP shows certainly there is value in labelling and arranging well-known behaviours into such concrete forms of *values*, *principles* and *practices* specifically aimed at solving a concrete problem. Those two practices work best when applied together, because of a synergy effect, as in the case of many other XP practices. The team leader can be more effective with a motivated team, while the team lead by a competent and smart individual will also do much better. This close relation is also emphasised by the integrity (I) factor of the DICE® framework.

Other agile practitioners are highly encouraged to contribute to this presented set of principles and practices to cover more and more of this unstable ground, as well as to empirically evaluate the ideas in different contexts.

4 Acknowledgements

The authors expresses their gratitude to the agile community for their feedback concerning the proposed collection of principles and practices. This work has been financially supported by the Ministry of Science and Higher Education, as a research grant 3 T11C 061 30 (years 2006-2007).

References

1. Williams, L., Maximilien, E.M., Vouk, M.: Test-Driven Development as a Defect-Reduction Practice. In: ISSRE '03: Proceedings of the 14th International Symposium on Software Reliability Engineering, Washington, DC, USA, IEEE Computer Society (2003) 34–48
2. Müller, M.M.: Are Reviews an Alternative to Pair Programming? Empirical Software Engineering **9**(4) (2004) 335–351
3. Madeyski, L.: Preliminary Analysis of the Effects of Pair Programming and Test-Driven Development on the External Code Quality. In Zieliński, K., Szmuc, T., eds.: Software Engineering: Evolution and Emerging Technologies. Volume 130 of Frontiers in Artificial Intelligence and Applications. IOS Press (2005) 113–123
4. Madeyski, L.: The Impact of Pair Programming and Test-Driven Development on Package Dependencies in Object-Oriented Design - An Experiment. In Münch, J., Vierimaa, M., eds.: Product Focused Software Process Improvement. Volume 4034 of LNCS., Berlin, Heidelberg, Springer (2006) 278–289
5. Hulkko, H., Abrahamsson, P.: A Multiple Case Study on the Impact of Pair Programming on Product Quality. In: ICSE '05: Proceedings of the 27th International Conference on Software Engineering, New York, NY, USA, ACM Press (2005) 495–504
6. Arisholm, E., Gallis, H., Dybå, T., Sjøberg, D.I.K.: Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. IEEE Transactions on Software Engineering **33**(2) (2007) 65–86

7. Madeyski, L.: On the Effects of Pair Programming on Thoroughness and Fault-Finding Effectiveness of Unit Tests. In Münch, J., Abrahamsson, P., eds.: Product Focused Software Process Improvement. Volume 4589 of LNCS., Springer (2007) 207–221
8. Madeyski, L.: Impact of pair programming on thoroughness and fault detection effectiveness of unit test suites. Software Process: Improvement and Practice (accepted), DOI: 10.1002/spip.382
9. Nosek, J.T.: The Case for Collaborative Programming. Communications of the ACM **41**(3) (1998) 105–108
10. Williams, L., Kessler, R.R., Cunningham, W., Jeffries, R.: Strengthening the Case for Pair Programming. IEEE Software **17**(4) (2000) 19–25
11. Nawrocki, J.R., Wojciechowski, A.: Experimental Evaluation of Pair Programming. In: ESCOM '01: European Software Control and Metrics, London (2001) 269–276
12. Müller, M.M.: Two controlled experiments concerning the comparison of pair programming to peer review. Journal of Systems and Software **78**(2) (2005) 166–179
13. Nawrocki, J.R., Jasiński, M., Olek, L., Lange, B.: Pair Programming vs. Side-by-Side Programming. In Richardson, I., Abrahamsson, P., Messnarz, R., eds.: Software Process Improvement. Volume 3792 of LNCS., Springer (2005) 28–38
14. Bloch, J.: Effective Java: Programming Language Guide. Addison-Wesley (2001)
15. Beck, K.: Test Driven Development: By Example. Addison-Wesley (2002)
16. Madeyski, L., Biela, W.: Empirical Evidence Principle and Joint Engagement Practice to Introduce XP. In Concas, G., Damiani, E., Paddeu, G., Scotto, M., Succi, G., eds.: Extreme Programming and Agile Processes in Software Engineering. Volume 4536 of LNCS., Springer (2007) 141–144
17. Sirkin, H.L., Keenan, P., Jackson, A.: The Hard Side of Change Management. Harvard Business Review **83**(10) (2005) 108–118
18. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change. 2nd edn. Addison-Wesley (2004)
19. Derby, E., Larsen, D.: Agile Retrospectives: Making Good Teams Great. Pragmatic Bookshelf (2006)
20. Schwaber K., Beedle M.: Agile Software Development with SCRUM. Prentice Hall (2001)