Agile Software Development Practices in Egypt SMEs: A Grounded Theory Investigation

Amr A. Mohallel, Julian M. Bass

Agile Software Development Practices in Egypt SMEs: A Grounded Theory Investigation

Amr A. Mohallel[1][0000-0002-0599-7853] and Julian M. Bass[2][0000-0002-0570-7086]

University of Salford, Manchester M5 4WT, UK
[1] a.hamed@edu.salford.ac.uk [2] j.bass@salford.ac.uk

Abstract. Agile information system development methods have been adopted by most software development organizations due to their proven benefits in terms of flexibility, reliability, and responsiveness. However, companies face significant challenges in adopting these approaches. Specifically, this research investigates challenges faced by software development companies in Egypt while transitioning to Agile. As little previous research is available targeting their concerns, we have conducted a grounded theory investigation. Key problem areas were found including lack of cadence in sprints planning, inadequate use of effort estimation and product quality issues.

The developed grounded theory reflects on the key problem areas found with SMEs adopting agile practices and can be used by software development practitioners adopting agile methods in Egypt or similar developing countries as an outline for the common problem areas they are expected to find.

Keywords: Agile Methods, Egypt, Agile Methods Adoption.

1 Introduction

Agile methods are based on an iterative and incremental approach where projects are divided into smaller analyze, implement, integrate, and test cycles. They have been shown to be more effective for software development than traditional waterfall models for small and large-scale projects [1].

Although Agile methods have been adopted by the software industry for more than a decade and started to rise with the Agile Manifesto in 2001 [9], it only started to gain popularity in Egypt a decade later. Based on the first author's previous experience of the Egyptian software development sector, we were aware that agile practitioners in Egypt are struggling to make the transition from traditional development models to Agile.

After conducting research interviews with Egyptian agile practitioners, we have learnt about their struggle with adopting agile development practices. Little previous research is available to study or investigate their problem areas, therefore, we are conducting this exploratory research to identify, evaluate, and potentially mitigate their problems.

We approached the problem using grounded theory which is based on interviewing agile practitioners and identifying their key problem areas. Findings from data collection were analyzed and evaluated in comparison to the current literate.

The aim of this research is to get an in-depth understanding of the agile software development practices and help Egyptian practitioners overcome their problem areas. This was achieved by interviewing 9 agile software development practitioners using Scrum, in 7 companies, in the north of Egypt. Scrum is the most popular method for agile project management [2] The findings showed mixed results in terms of agile adoption.

Although the interviews did show clear motivation from practitioners to adopt agile methods, four key common problem areas were found:

- Scrum is based on *Sprints* i.e. repeatable time-boxes during which a potentially shippable product is delivered. Sprints ideally vary from 1-4 weeks [2]. The reason why these time-boxes are fixed is that they help teams calculate their *Velocity.* Velocity is one of the key outcomes of applying Scrum to software development. It is a metric for work done by a team in a sprint. Scrum teams use velocity for effort estimation before a project starts and for forecasting the amount of time needed to complete a given project [16]. We discovered that companies tend to change the length of sprints based on the workload they put in each.
- Unlike traditional software development methods that calculate the time needed to complete a given task by looking at how big it is, Scrum uses story points which takes priority, size and complexity into consideration. We discovered that companies have insufficient use of story points which leads to a dramatic failure in sprints planning.
- As Scrum promotes constant deliverables in short periods of time to enhance, practitioners experience higher levels of stress which negatively affect the quality of products.
- Teams have difficulties handling task switching and handoffs.

The contribution of this research is a grounded theory that can be used by software developers adopting agile methods in Egypt or similar developing countries that sets an outline of the common problem they are expected to find and how to mitigate them.

2 Related Work

Software development models have been produced to help organize, scope and keep software projects on time and within budget. Agile software development methodologies were developed to help solve the problems aroused by traditional development methodologies [3]. The main goal of applying Agile methods is having a more adaptable, flexible and responsive development lifecycle.

It has been shown that agile methods improve both product quality and development productivity [21], they enhanced customer involvement, adaptability and incremental delivery of software projects [22].

The Agile manifesto defined a set of values and principles for developing software projects [20]. Since then, several methods (e.g. Lean [23], Extreme Programming [9] and Scrum [23]) were defined to provide guidelines and clear up the application of the Agile values in the development of software projects.

Agile development is based on self-organizing teams [26] who collaborate to adjust to customers' changing requirements [24]. Several practices are adopted in Agile methods, however, daily stand-ups, sprint planning and retrospectives are the most commonly used, with a percentage of 90%, 88% and 85% respectively [6].

Although guidelines and practices are defined for applying Agile methods in software development, companies, especially SMEs tend to 'cherry-pick' selected Scrum and XP practices from the full constellation of practices available [19].

Agile software development has been an active area of research ever since the agile manifesto was published in 2001, however, the research literature fails to reflect specific challenges faced by Egyptian practitioners. By 2012, from a sum of 1,427 papers on agile methodologies, only one paper was based on Egyptian experience [25].

Some research did focus on investigating software development challenges faced by practitioners in smaller software companies in specific regions e.g. sub-Saharan Africa where contributions were to literature in areas such as agile awareness and adoption challenges [8]. It was concluded that further research is required on how agile methodologies can be tailored for use in a developing country context.

3 Research Methods

3.1 Research Sites

We were curious about exploring the transition process and whether agile is adding value to the practitioners or not. There is a lot of focus in literature on studying agile practices in European/western countries, while little research focused on studying agile practices in Egypt [25]. In that sense, we do believe that this study is groundbreaking.

3.2 Data Collection

As presented in [4], this study is to follow a grounded theory approach to gather and analyse data from agile practitioners in Egypt. Grounded theory is a qualitative research method that seeks to discover emerging patterns in data. Data collection was based on reviewing agile practices in literature, artefact modelling and interviewing agile practitioners.

Sampling. The sample identified in this study is composed of 9 agile practitioners including 3 project managers, 1 business analyst (ex-developer), 1 quality assurance engineer, 1 senior software engineer and 3 junior software developers. The selected companies have 5-25 employees with businesses offering custom web-based software solutions. All 7 companies have been using Scrum for less than 5 years.

After engaging 9 interviewees, the data collection process did reach a saturation level as no new data is being reviled by collecting more data if more interviewees are to be made. Qualitative research should not be concerned about how many people contributed to a study as long as the interviews did discover new constructs and values [5].

Although minor data will always be reviled from conducting more interviews, the main issues raised as a result from the interviews were pretty much the same (as explained further). For this reason, the chosen individuals who contributed to the study were chosen carefully to represent different points of views and aspects depending on their various specialties and experiences dealing with practicing agile development.

Coding Scheme. Companies were coded from Comp1 to Comp7, while participants were coded as follows:

- Project managers: PM1, PM2, and PM3
- Business analysts: BA1
- Senior software engineers: SSE1
- Quality assurance engineers: QA1
- Junior software developers: JD1, JD2, and JD3

Creating the Interview Guides. As Scrum was initially based on principles from Lean manufacturing [7], and since the fact that the main idea behind Lean is to provoke the customer-focused value idea while mitigating waste as is the case for all the agile approaches, the interviews were built to help rate each company on how Lean they are. The interview guide was developed to cover each individual's implementation of agile practices and artefacts along with their use of Lean software development.

1.3 Data Analysis

The data is to be extracted from the interview scripts following a content analysis approach [26]. In content analysis, responses are coded based on patterns or themes found in the interview scripts.

Selective Coding (also known as *Open Coding*) is the first step of data analysis where it starts by noting down *Key Points* during and post every interview, then coding these points to summarize findings [26].

While conducting the interviews, memos were written down to summarize the main issues faced by each interviewee. Then the constant comparison method [26] was followed where the memos were constantly edited and enhanced. By the end of the interviewing process, the core issues raised were categorized into core categories based on the generated memos.

4 Findings and Data Analysis

While conducting the interviews, key points were noted along with a memo for each interview reflecting on the key problem areas faced by each individual. The constant comparison method [26] was used to enhance and develop the resulting memos. After having all the memos written, three of them were chosen to represent three core categories. The resulting core categories are:

1.4 Lack of Cadence in Sprints Planning

The conducted interviews did investigate the efficiency of the teams' planning for their sprints and backlog items. The first interesting topic raised was the dynamic duration of the sprints. 4 out of the 7 companies involved in the study did plan for dynamic sprints where a sprint duration is decided mostly upon the amount of work that needs to be done in the given sprint.

All participants were asked about their default sprint length and whether they change it or not. Table 1 summarizes their answers.

Table 1. - A table showing for each company, the sprints duration and if they're dynamic or not

Companies	Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7
Participants	PM1&BA1	SSE1	JD1&JD2	JD3	PM2	PM3	QA1
Duration (weeks)	2-4	1.5	4	2	3	1-4	2-3
Dynamic?	Yes	Yes	Yes	Yes	No	Yes	No

As presented in table 5.1, we had different opinions on the lengths and dynamic nature of the sprints. *"Some sprints were 2 weeks, others were 3, and sometimes it could take a bit longer to a month. Sometimes we have very short sprints, as for those made to fix bugs"* -PM3; *"Sprints length changes. The first two sprints were 3 weeks. The current sprint is four"* -PM1; *"95-98% of our sprints are fixed to 3 weeks. Under some circumstances, we could have a hotfix, or we find that a group of story points can better be completed in the same sprint together, so we could have a 1-2 weeks sprint to get them all done then*." –PM1.

So it appeared that most of the companies do change the duration of their sprints depending on the workload in each sprint. This has been described by using terms such as "Well, it depends on what are we working on" -JD1; "So you decide a sprint length depending on the amount of user stories you want to accomplish in it?"-Interviewer, "Basically, yes"- PM1.

Comp2 seemed to have shorter sprints to get faster feedback; "We 'tend' not to exceed one and a half weeks. Just not to redo a lot of work if the feedback wasn't positive" -SSE1.

1.5 The Inadequate Use of Effort Estimations

After investigating the sprints durations, all participants were asked to rate their sprint planning from their own point of view as follows:

> *"How much would you rate your sprint planning in a scale from 0-10? Where 0 reflects very bad planning, and 10 reflects perfect planning."* -Interviewer

As presented in table 5.2 below, the interviewees gave an average rate of 4.5 to the sprints planning of their current or last completed projects.

Table 1. - Interviewee's individual rating on their company's sprints planning efficiency

Interviewee	PM1	PM2	PM3	BA1	SSE1	QA1	JD1	JD2	JD3
Rating	7	8	7-8	0	3	7	3-4	-	5

Most participants mentioned that the lack of efficiency of sprints planning is mostly related to weak effort estimation. This was clear to us from answers like "[Planning fails] Mainly due to the bad estimation of time"-JD1; "[Sprint planning is] Very bad mostly due to bad estimates" –BA1; "Mainly due to the bad estimation of time"-JD1; "I believe estimation is the most painful part"-PM2.

While the inaccurate effort estimation problem started being a key point, we started investigating the estimation criteria in each company. To begin with, the interviewees were asked about the units they use for measuring their estimates. Five companies were using points-based estimation as their used effort estimation unit, while 2 were using hours.

Participants who used hourly-based estimation agreed on its unreliability; "I believe time-based estimations always give false numbers" -BA1; "I believe estimating tasks in hours just makes everything stressful. It doesn't usually count time for testing, validation, documentation, etc." -JD2.

However, those some of those who used story points mentioned that they struggle with applying it. "I wasn't comfortable using it. Neither me, nor my teams."-PM3; "see it beneficial when it comes to performance tracking" -QA1. PM2 stated that story points were "easy to understand, but definitely hard to master". Four out of the five companies using story points clearly shown the teams misunderstanding of how points estimation works. "Developers always tend to use time-based system. They normally calculate how much time would it take them to finish a given task, then convert the hours to points" -BA1. SSE1 explained how they estimate points as "We use points where each point represents two hours.

Although PM1 did mention how he does not encourage developers to use time as a main factor when deciding how many points they need to complete a given task, his opinion was still somehow unclear. So he was asked if a developer has a min or max number of points to burn in a given period of time. He answered "Typically, a developer should burn an average of 2 story points per day. A working day has 7 working hours; therefore, a story point should take an average of 3.5 working hours." –PM1.

PM2 made a point on the connection between time and points showing how important the time factor is as this is what the client wants to hear at the end (similar to PM3). He also added how he does intrude the estimation process when he finds out that his development teams are giving estimations that would not fit with budget. Seems like it was the case with JD2, she stated that she gets assigned to a task with an estimated time to finish it. Same with BA1 where he said that "Team leaders make a point estimation and pass it to developers." –BA1. PM2 does the same with his team, and when asked for the reason, he said "When you give a certain team the full responsibility to make their own estimations, they always tend to over-estimate story points just to feel more relaxed and less tensed while developing the given project." –PM2

1.6 The Pressure Factor and Code Quality

While the constant pressure on software development teams was being mentioned by more participants, a direct connection started to rise between the pressure and the overall product quality.

As claimed by PM2 in the quote above, teams do tend to over-estimate the efforts they need to finish a given job. He claimed that the reason behind their tendency to over-estimate is "just to feel more relaxed and less tensed".

JD1 and 2 did show their concern with agile being stressful; "[Scrum] is generally more stressful"; she later added "it stresses me out having to attend a meeting every day to report my progress" -JD1. PM3 also made the same point on stress stating:

"Instead of dividing a given system into phases and get feedback on each phase, we had to develop fully-functional releases that the customer can actually use and start inputting data into. This caused more much stress on the development team due to the small intervals of time needed to produce functional releases, however, the teams produced better outputs" -PM3.

QA1 approved the point stating that he has issues with Scrum rushing their work; "We always have pressure and deadlines where we have to submit tasks and so on, sometimes there are too many stories are assigned to a two weeks sprint. Sometimes we don't have time to finish them while maintaining testing and so on." –QA1"

JD2 mentioned that her main cause of stress is that "..They [the team leaders who estimate the tasks for her] normally do not count the time needed for testing, validation, documentation, etc. So, when I have a short period of time, I find myself skipping everything but actual coding." –JD2

Those participants how mentioned that Scrum makes them to be in a constant rush, they were asked about how they think rushing the project affect the quality of their work.

Some participants mentioned this affects documenting their code; "We have no time to document the code." –SSE1; "I never really have time [for documentation]"-JD2; "As the nature of work in the company being in a continuous rush; documentation is often done." –QA1.

Others mentioned problems with testing; "I believe it causes a lot of pressure on us. Rushing always prevents us from having enough time to conduct complete regression testing to discover problems or bugs." -QA1; "Due to the fact that he [the manager] is always in a rush, we couldn't make any test-driven development or unit testing, etc." –JD3

5 Discussions and Evaluation

The data collection did show how the participants were motivated to adopt the agile project management methodologies, more precisely Scrum. All the participants did show decent understanding of the agile practices and how they could be applied.

1.7 Key Problem Areas

Inadequate Use of Effort Estimation. The accuracy of effort estimation can indeed determine the success or failure of any given software development project [10]. As presented in the data analysis, participants did manage to find the connection between weak effort estimation and weak sprints planning. In fact, some practitioners did mention how effort estimation is their biggest problem since they made the transition to adopting the agile methodologies.

While analyzing the collected data, we found out that five out of the seven companies that participated in the study did use the Scrum point-based estimation system which is the recommended effort estimation system in Scrum [11] and it is the most used estimation system in Scrum practitioners worldwide [12], nevertheless, practitioners did find difficulties using it.

The estimation accuracy for teams do increase remarkably when planning poker is used for planning releases [13]. Two studies were conducted to compare a given group's estimation accuracy using planning poker to traditional individual-expert-based estimations. The results did show how the group's estimations using planning poker were much less optimistic and much more realistic [14].

The story points estimations does rely on three factors i.e. priority (=urgency*business value), size, and complexity factors [15], however, as presented in the analysis above, teams always tend to use time as the only factor when estimating their efforts. Although some research does not include time as a factor at all [15], PM2 did mention how his teams realized that time is at least becoming a less important factor by time. When he was asked about his teams' accuracy with estimating efforts using points, he did mention how their accuracy was raised from 60% to 70% in one year. QA1 also mentioned how the more his team work together, the more accurate estimations they make.

Estimating points using planning poker can be difficult, but our findings and literature does prove how teams show constant improvement as they do more work together. In case the teams are given tasks with pre-estimated points, there will be no chance of getting the estimation accuracy increasing.

The Lack of Cadence in Sprint Planning. Although the teams' commitment to their sprint duration is a crucial pillar of Scrum [18], Our findings along with cases in literature [18] did show how development teams struggle to stay committed to their sprint length decisions, so they end up extending it to fit the required user stories. In fact, some participants did build the whole plan on a dynamic sprints basis.

Calculating the velocity, which is the key metric of Scrum [16], is based on having the sprints duration fixed. Without velocity, it is impossible for a product owner to estimate how many sprints a team needs to burndown a given number of stories. However, this didn’t seem to be a problem for most of the participants. We had interesting opinions about the reason why they do not worry about velocity; “each developer/ team builds a reputation by time that indicates whether he/they can finish the assigned task in the given story points they estimated. I don’t believe team velocity should be calculated as a quantitative number.” –PM1; “I don’t bother [calculating velocity]. I don’t find it [burndown charts] very beneficial if the team is going smooth with the estimates” -PM3.

Constant Pressure and Code Quality. As presented in the findings about, all the participants -apart from project managers- did state that they either do minimal testing and almost no documentation for their code. The findings did explain how the main reason was the constant pressure and lack of time availability.

Many reasons can cause pressure on the team; that is why it is the scrum master’s responsibility to be constantly observing his team’s social/psychological aspects. Product owners or the management can also indicate whether the team is under pressure or not.

The findings did show the constant problem between project managers not relying completely on a team to make his estimations because they are expected to over-estimate their effort to be more relaxed, and on the other hand, developers do need time to make sure they get their job done right.

In Scrum, although the development teams can make use of some experienced guidance, they must estimate the story points for each backlog item, and they should be responsible for choosing how much work they can do in each sprint [17]. However, in some investigated cases, the development teams were not trusted to give their own estimations. They were either given a task with pre-estimated duration or given a certain amount of tasks to complete in a given sprint.

Agile and Flexibility. Sprints in Scrum should range from one to four weeks having the same length during the project [2]. The term “suggested” was intentionally used in the above sentence. In fact, most of the Scrum practices are not standardized. There are no rules or guidelines that can walk you through some steps or practices that can guide you to make the best use out of agile. It is seen more like a framework. It is based on some set of principles providing the foundation to which a team will add its own approaches and practices. The result of applying Scrum will always be a unique version for each development team.

However, practitioners did seem to confuse the difference between customizing a practice and ignoring it. Scrum does offer some foundational practices where the more you ignore them, the less benefits you are getting from the whole approach.

The term “suggested” did not refer to the sprint length being fixed. It was proven that the companies using dynamic sprints are totally losing the benefits of calculating velocity, hence not being able to make accurate forecasting, hence making the burndown chart lose its indication capabilities.

Although the more teams collaborate, the more they realize that burndown charts are getting more and more accurate, and the velocity is stabilizing over time. Scrum is not about steadily improving the velocity, it’s about making the velocity measures more accurate, so that you can more reliably predict how much you can develop in an iteration. Scrum teams work more on driving predictability in the process more than they work on productivity.

6 Conclusions

The participants in our research mentioned how adopting agile had a positive reflection on both their software development process and the overall satisfaction of their customers. However, since their migration to agile, many challenges were faced to implement Scrum, the most commonly used method for agile project management.

This paper presents a grounded theory investigation that was conducted with agile practitioners in Egypt regarding the challenges they face with adopting agile methods. Memos were written to reflect on the main problems faced by practitioners. Using the constant comparison methods, core categories were written to represent the common key problem areas found during the memoing process. These problem areas are (a) lack of cadence and insufficient planning of sprints, (b) constant pressure on development teams and (c) inadequate use of effort estimation.

7 Future Work

The next stage of our research will be to explore lean principles e.g. eliminating waste, amplifying learning, team empowerment, and building integrity in. We expected to find companies using Lean principles, so we built our interviews to reflect on how *Lean* companies are. However, our approach did not fit very well as what we found the challenges to be more fundamental with applying agile development which made the whole assessment process in terms of Leanness not practical.

References

1. Brhel, M., Meth, H., Maedche, A., Werder, K.: Exploring Principles of User-Centered Agile Software Development. Information and Software Technology 61(C), 163-181 (2015).
2. Rubin, K.: Essential Scrum: A practical guide to the most popular Agile process. 1st edn. Addison-Wesley Professional (2012).
3. Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., Still, J.: The impact of Agile practices on communication in software development. Empirical Software Engineering, 13(3), pp. 303–337 (2008).
4. Walker, D., Myrick, F.: Grounded theory: an exploration of process and procedure. Qualitative Health Research. 16(4), pp. 547-559 (2006).
5. Steinberg, W. Price, M.: Statistics alive!. 2nd edn. Sage Publications, Los Angeles, USA (2011).
6. Rahy, S., Bass, M. J.: Information flows at inter-team boundaries in agile information systems development. In: 15th European, Mediterranean and Middle Eastern Conference on Information Systems (EMCIS). Limassol, Cyprus (2018)

7. Janes, A.: A guide to lean software development in action. In: Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference. (2015).
8. Regassa Z., Bass J.M., Midekso D.: Agile Methods in Ethiopia: An Empirical Study. In: Choudrie J., Islam M., Wahid F., Bass J., Priyatma J. (eds) Information and Communication Technologies for Development. ICT4D 2017. IFIP Advances in Information and Communication Technology, vol 504. Springer, Cham (2017).
9. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999).
10. Akhtar, N., Ghafir, S., Tripathi, S.: Effort Estimation of the Scrum based Software Projects using Particle Swarm Optimization. Advances in Computer Science and Information Technology (ACSIT), 2(7), pp. 24-26 (2015).
11. Gandomani, T., Wei, K. and Binhamid, A.: A Case Study Research on Software Cost Estimation Using Experts' Estimates, Wideband Delphi, and Planning Poker Technique. International Journal of Software Engineering and Its Applications, 8(11), pp.73-182 (2014).
12. Mahnic, V.: A case study on agile estimating and planning using scrum. Elektronika, 111(5), pp.123-128 (2011).
13. Mendez-Fernandez, D., Penzenstadler, B., Kuhrmann, M. and Broy, M.: A Meta Model for Artefact-Orientation: Fundamentals and Lessons Learned in Requirements Engineering. In Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems (MODELS 2010), 6395/2010, pp.183–197 (2010).
14. Usman, M., Mendes, E., Britto, R., Weidt, F.: Effort estimation in agile software development. In Proceedings of the 10th International Conference on Predictive Models in Software Engineering - PROMISE. (2014)
15. Zahraoui, H. and Idrissi, M.: Adjusting story points calculation in scrum effort & time estimation. Intelligent Systems: Theories and Applications (SITA), 2015 10th International Conference (2015).
16. Downey, S., Sutherland, J., Scrum Metrics for Hyperproductive Teams: How They Fly like Fighter Aircraft. In: 46th Hawaii International Conference on System Sciences. IEEE, Wailea, USA (2013).
17. Viscardi, S.: The Professional ScrumMaster's Handbook. Packt Publishing (2013).
18. Sutherland, J. and Schwaber, K.: The Scrum Guide. O'reilly (2013).
19. Clutterbuck, P., Rowlands, T., Seamons, O.: A case study of SME web application development effectiveness via agile methods. Electron. J. Inf. Syst. Eval. 12(1), 13–26 (2009).
20. Agilemanifesto, http://agilemanifesto.org/, last accessed 10/10/18.
21. Dyba, T., Dingsøyr, T.: What do we know about agile software development? IEEE Software. 26(5), pp 6–9 (2009).
22. Dingsoyr, T., Nerur, S., Balijepally, V., Moe, N.B.: A decade of agile methodologies: Towards explaining agile software development (2012).
23. Poppendieck, M.B.: Lean software development: an agile toolkit. London: Boston Mass. Addison-Wesley, London (2003).
24. Santos, V., Goldman, A., Desouza, C.: Fostering effective inter-team knowledge sharing in agile software development. Empirical Software Engineering, 20(4), pp. 1006-1051 (2015).
25. Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N. B.: A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, 85(6), 1213–1221 (2012).
26. Hoda, R., Noble J., Marshall S.: Supporting Self-organizing Agile Teams. In: Sillitti A., Hazzan O., Bache E., Albaladejo X. (eds) Agile Processes in Software Engineering and

Extreme Programming. XP 2011. Lecture Notes in Business Information Processing, vol 77. Springer, Berlin, Heidelberg (2011).