AGILE SOFTWARE DEVELOPMENT METHODOLOGIES: AN OVERVIEW OF THE CURRENT STATE OF RESEARCH

Năftănăilă Ionel
University of Economic Studies (ASE) Bucharest Faculty of Management Piaţa Romană 6, Bucharest, Romania ionel@naftanaila.ro +40213191900

Under the current economic conditions many organizations strive to continue the trend towards adopting agile processes, in order to take advantage of the numerous benefits that these can offer. Those benefits include quicker return on investment, better software quality, and higher customer satisfaction. To date, however, there is no structured body of research that can guide organizations in adopting agile practices. To address this situation, the current paper identifies and structures the main theoretical contributions to the field of Agile Methodologies research, by presenting Agile Methodologies, by analyzing the main papers on social implications of using Agile, by presenting the main studies on implementation of Agile and by synthesizing the research with regard to communication in Agile projects.

Keywords: Agile Software Development, Scrum, Project Management Methodologies, eXtreme Programming

JEL code:M10

Introduction and motivation

Since the beginning of the current world financial crisis many technology-driven companies have suffered the effects, being forced to lay off people or drastically diminish costs (Wauter, 2009). The survival of the company itself becomes dependant of the time-to-market, deliver on time to the customer and minimize costs. The scientific literature abounds of examples in which the success of projects drive the success of companies, or, the other way around, the failure of a project puts the company out of business (Charette, 2005), (Voas & Whittaker, 2002), (Jones, 1995). As a consequence, minimizing risk and approaching projects in a structured manner become critical success factors. Over the past few years software development organizations have learned about the benefits of Agile Methodologies, such as Scrum and XP. The scientific literature and business journals present numerous success stories highlighting the benefits of organizations which successfully adopted agile practices. As a result, many organizations are now aspiring to adopt agile practices.

Overview of Agile Methodologies

Agile represents a group of software engineering methodologies which promise to deliver increased productivity, quality and project success rate overall in software development projects. Such methodologies are SCRUM (Schwaber & Beedle, Agile Software Development with Scrum, 2001), XP (Beck & Andres, Extreme Programming Explained: Embrace Change, 2004), or the lesser-known Crystal (Cockburn, 2001). The outline of Agile Methodologies was laid down by the Agile Manifesto, published by a group of software practitioners (Beck et. al, 2001).

Scientific literature on the subject (Highsmith, 2002) suggests that the differences between traditional methodologies and Agile Methodologies relies on two main assumptions: First, traditional methodologies assume that customers do not know their requirements, hence they need guidance from the developers, but Agile Methodologies assume that both customers and developers do not have full understanding of requirements when the project starts. Therefore, in traditional software development environments, developers want a detailed specification, whereas in Agile Methodologies customers and developers learn together about the system requirements as the development process evolves. Second, traditional methodologies assume that customers' ability to foresee their future requirements is limited, and as such developers have to build in extra functionalities to meet these future needs, often leading to overdesigned system. On the other hand, Agile Methodologies emphasize simplicity.

Research on social implications of using Agile Methodologies

All Agile Methodologies have in common a certain emphasize on social aspects of software development, taking in consideration a series of explicit values, such as communication (Schwaber & Beedle, Agile Software Development with Scrum, 2001), or courage (Beck & Andres, Extreme Programming Explained: Embrace Change, 2004). Also, these methodologies involve a set of best practices such as pair programming, continuous integration or daily deployments (Beck & Andres, Extreme Programming Explained: Embrace Change, 2004).

It is found in the literature that usually the mature agile teams have a better social and technical cohesion, as close communication is crucial to success (MacKenzie & Monk, 2004). Code writing is often performed by pairs of developers, while the traditional roles in software development (analyst, tester, and architect) disappear (Năftănăilă, 2008).

While the traditional waterfall methodologies rely on a large number of documents and artefacts, SCRUM and the rest of Agile Methodologies use a minimum of documentation, just sufficient for the project to run under good

conditions. Two of these artefacts are the story cards and the wall (Sharp, Robinson, & Petre, The role of physical artefacts in agile software development: Two complementary perspectives, 2009), which bear two main purposes: enable the capturing of requirements and support the development process. It has been shown, though, that different teams use slightly different conventions for these artefacts, in terms of card layout, card colour and organising principles for the wall. There are, however, some common traits. For example, most teams' user stories follow are written in natural language, and use the widely known template proposed by XP (Beck & Fowler, Planning Extreme Programming, 2000), following the pattern "As a <<role>> I want <<behaviour>> so that <<benefit>>". Each card usually gets through a general life-cycle such as: story is written on the card, card is prioritized by the customer, card is estimated by the team, card is assigned to an iteration, implemented by developers, and accepted as "done" (Schwaber, SCRUM Development Process, 1995).

The wall is usually a whiteboard (in SCRUM) where team members display the user stories, and which they use as a visual "control panel" of the project. Beside the user stories, the wall usually holds other items, such as the burndown chart (measure of project's evolution). Daily stand-up meetings involved by the Agile Methodologies take place around the wall.

The study of (Sharp, Robinson, & Petre, The role of physical artefacts in agile software development: Two complementary perspectives, 2009) aims to shed light over using these rather simple artefacts in Agile Methodologies (and implicitly in Scrum) by analyzing the notational and social perspectives of using the story cards and the wall. The authors find that, besides the fact that both the users stories and the wall have their own separate meanings, they have strong *combined* meanings (meanings that occur only when they are used together). Therefore, from a *notational perspective* authors find that using the story cards and the wall leads to: *closeness of mapping* between the users' minds and what they are trying to express, appropriate level of *abstraction*, providing means of *secondary notation* (ex. by using colours, layout and labels), bringing *consistency* in the project, reduce *diffuseness* of meaning, show *hidden dependencies*, and improve overall *visibility* in the project. From a *social perspective*, authors infer that notation does not exist in isolation, it has to be situated in the reality of a social setting. For instance, the authors show that in relation to the story cards, the teams have developed great care and respect (especially when handling them physically, in situations like moving them on the wall, or annotating them). Authorship becomes meaningful, while handwriting and initials become a form of a secondary notation. In the same time, the wall becomes centric for the social life of the team – mediates and manages the life of developers. The position of the card on the wall becomes highly significant – when a card is moved to the "done" area, developers feel professional satisfaction and achievement.

The implications of their article, especially when implementing Agile Methodologies, are that using software tools to manage Agile teams can be done, but only after a social context and meaning has been created. There are reports of situations when teams have two means of storing the artefacts: physically for collocated team members, and in a software tool, for remote team members. However, in order to maintain the physical significance of artefacts, the teams have backed-up the software tool with phone conversation and even photos of the wall at critical development stages (Sharp, Customer collaboration in distributed agile teams, 2008).

Research on implementation of Agile Methodologies

In terms of *implementation* of Agile Methodologies, the literature is rather scarce. We can identify the study of (Nerur, Mahapatra, & Mangalaraj, 2005) who show that migrating to Agile Methodologies involve issues regarding management, people, technology and process.

In terms of acceptance of Agile Methodologies, we can identify the significant study of (Chan & Thong, 2009) which attempts to address what can be done to overcome the challenge to Agile Methodologies acceptance. They provide a critical review of the extant literature on the acceptance of traditional SDMs and Agile Methodologies, and develop a conceptual framework for Agile Methodologies acceptance based on a knowledge management perspective.

Based on previous work on Agile Methodologies (mostly case studies) in papers such as (Ceschi, Sillitti, Succi, & Panfilis, 2005), or (Cohn & Ford, 2003) they propose a conceptual framework regarding the acceptance of Agile Methodologies. They propose that a series of factors, such as (a) Ability-related factors (Self efficacy, Experience, Training, External Support), (b) Motivation-related factors (Career consequences, Top Management support, Voluntariness, Subjective norm, Organizational culture), (c) Opportunity-related factors (Teamwork, Communication, Shared understanding, Arduous relationship) influence Knowledge Management Outcomes (Knowledge Creation, Retention and Transfer). Knowledge Management Outcomes, on the other hand, along with Agile Methodology characteristics (Perceived usefulness, Perceived ease of use, Perceived compatibility, Perceived demonstrability, Perceived maturity) lead to Acceptance.

While this framework is yet to be empirically proved solid, it can be considered significant because it brings knowledge management as another perspective in examining acceptance of software development methodologies, on one hand, and because it synthesizes and critically analyses the previous literature on this subject.

Another paper proposing a framework for implementing and improving Agile Methodologies in practice is the one of (Qumer & Henderson-Sellers, 2008). The authors depart from the hypothesis that in practice, few organizations

are able to take on an agile development approach immediately and adopt them successfully over a short period of time – in many cases, a full implementation requires years. The authors present and explain the Agile Software Solution Framework (ASSF). While testing the model proposed, the authors prove (although this is a collateral finding of their study) that SCRUM presents the highest *degree of agility*[197] in terms of practice. The Agile Software Solution Framework (ASSF) proposed by them can be used to create, modify or tailor situation-specific agile software by using a situational method engineering approach, feedback and a standard meta-model. The authors have embedded a number of new models and processes in ASSF, such as an agility measurement model and process, an agile adoption and improvement model and process, an agile software solution framework knowledge-base engineering and management process, an agile workspace and an Agile Toolkit.

Studies on communication in Agile Projects

In previous literature on software project management it has been shown that *communication* is an important success factor (Stelzer & Mellis, 1998), and that communication is considered to make software development more efficient in companies (Paasivaara & Lassenius, 2003). The main problem seems to emerge from the fact that all the players in the software development process (users, customers, team, maintenance team, management) view and communicate regarding the same product from different perspectives: users require the product to have a large degree of usability, customers seek reliability and low maintenance costs, as well as fast time-to-market, managers seek minimizing costs, maintenance teams seek documentation and reliability, while the development team seeks technical challenges and moving to the next project (Boehm & Ross, 1989).

Communication, on the other hand, is not very well covered with regard to Agile Methods in general, and with SCRUM in particular. The paper of (Pikkarainen, Haikara, Salo, Abrahamsson, & Still, 2008) aims to increase the understanding of communication in the context of agile software development: internally among the developers and project leaders and in the interface between the development team and stakeholders.

Their study (under the reserve of being a qualitative research, based on two case studies carried in the same organization) shows a few interesting conclusions in terms of communication in projects. For instance, SCRUM puts a great emphasis on the daily stand-up meetings, which are in general perceived as being helpful in reducing the confusion about what should be developed (Mann & Maurer, 2005). On the other hand the authors, referencing the work of (Murru, Deias, & Mugheddue, 2003), suggest that the sprint planning meetings might cause the risk that the most demanding customers get what they want and are favoured by this approach, but the decisions are not always analysed in enough detail from a technical perspective, leading implicitly to a negative impact on the overall project goal. Another example refers to the open space concept: although generally the open-space environment is perceived as increasing productivity in software development, the authors have found evidence of situations when open-space setting was perceived as being distractive. Also, when it comes to the communication between the project team and the project stakeholders (customer, management, etc), the conclusions of (Turner, 2003) with regard to the fact that limited formal and informal
communication mechanisms can hinder communication between pilot project teams in
the context of agile software development have been confirmed by the authors. The work of (Rising & Janoff, 2000) is also confirmed by (Pikkarainen, Haikara, Salo, Abrahamsson, & Still, 2008), supporting the hypothesis according to which short time-boxed iterations in agile software development are key reason for improving communication in software development teams.

Empirical studies regarding Agile Methodologies

It has been argued that the current research has only a few case studies on agile software
Development (Layman, Williams, & Cunningham, 2006). While this is true, there are a few papers which empirically study Agile Methods, such as Scrum or XP. For instance, the paper of (Salo & Abrahamsson, 2008) brings a series of interesting findings from an empirical study of Scrum and XP in European embedded software development organizations; for instance, the authors show that 77% of the respondents who have used Scrum have reported positive experiences, while 27% of the respondents claim to use Scrum systematically or mostly through the project. However, the number of empirical studies remains low, and further research must to be conducted in this area.

Conclusions and further research

Agile Methodologies are more and more used in software development companies; even large companies, such as Microsoft, have started to use it – which shows increasing importance as well as increasing recognition of this group of methodologies.

The paper analyses the current state of research with regard to Agile Methodologies. Although many articles and papers have been published, only a few represent significant empirical papers, most of them being case studies and anecdotic evidence. Therefore, there is a strong need for more empirical studies in this field; from this perspective,

197 Determined using their Agility Calculator known as the 4-DAT which measures the key attributes of agility: flexibility, speed, leanness, learning and responsiveness.

the current paper can constitute the departure point, as it synthesizes and structures the most significant body of research to-date. From a practitioner's perspective, the current paper can be used as a departure point in implementing Agile Methodologies, by providing a comprehensive synthesis of the most significant sources of practical knowledge.

While one of the conclusions that can be drawn from the above analysis is that without doubt using Agile Methodologies brings substantial benefits to the companies, the current paper also shows that the current state of research lack of studies which analyses use and implementation of agile practices in teams and organizations.

Bibliography

1. Beck et. al. (2001). Manifesto for Agile Software Development. Retrieved 05 01, 2009, from agilemanifesto.org: http://agilemanifesto.org/
2. Beck, K., & Andres, C. (2004). Extreme Programming Explained: Embrace Change. Addison-Wesley Professional.
3. Beck, K., & Fowler, M. (2000). Planning Extreme Programming. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
4. Boehm, B., & Ross, R. (1989). Theory-W software project management principles and examples. IEEE Trans Software Engineering , 15 (7), 902-916.
5. Ceschi, M., Sillitti, A., Succi, G., & Panfilis, S. (2005). Project management in plan-based and agile companies. IEEE Software , 22 (3), 21-27.
6. Chan, F., & Thong, J. (2009). Acceptance of agile methodologies: A critical review and conceptual framework. Decision Support Systems (46), 803-814.
7. Charette, R. N. (2005, 09). Why Software Fails. Retrieved 3 14, 2007, from IEEE Spectrum: http://www.spectrum.ieee.org/sep05/1685
8. Cockburn, A. (2001). Agile Software Development. Addison-Wesley Professional.
9. Cohn, M., & Ford, D. (2003). Introducing an agile process to an organization. IEEE Computer , 36 (6), 74-78.
10. Highsmith, J. (2002). Agile Software Development Ecosystems. Addison-Wesley Professional.
11. Jones, C. (1995, 3). Patterns of large software systems: failure and success. Computer , pp. 86-87.
12. Layman, L., Williams, L., & Cunningham, L. (2006). Motivations and measurements in an agile case study. Journal of Systems Architecture , 654-667.
13. MacKenzie, A., & Monk, S. (2004). From cards to code: How extreme programming re-embodies programming as a colective practice. COMPUTER SUPPORTED COOPERATIVE WORK , 13 (1), 91-117.
14. Mann, C., & Maurer, F. (2005). A case study on the impact of SCRUM on overtime and customer satisfaction. Proceedings of the Agile Conference, (pp. 70-79).
15. Murru, O., Deias, R., & Mugheddue, G. (2003). Assessing XP at a European Internet Company. IEEE Software , 37-43.
16. Năftănăilă, I. (2008). ANALELE UNIVERSITATII DIN ORADEA, STIINTE ECONOMICE , 4 (8), 435-440.
17. Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. Communications of the ACM , 48 (5), 73-78.
18. Paasivaara, M., & Lassenius, C. (2003). Collaboration Practices in Global Inter-Organizational Software Development Projects. SOFTWARE PROCESS IMPROVEMENT AND PRACTICE , 183-200.
19. Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development. Empir Software Eng , 13, 303–337.
20. Qumer, A., & Henderson-Sellers, B. (2008). A framework to support the evaluation, adoption and improvement of agile methods in practice. The Journal of Systems and Software , 81, 1899-1919.
21. Rising, L., & Janoff, N. (2000). The SCRUM software development process for small teams. IEEE Software , 21-28.
22. Salo, O., & Abrahamsson, P. (2008). Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum. IET Software , 58-64.
23. Schwaber, K. (1995). SCRUM Development Process. Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA), (pp. 117-134).
24. Schwaber, K., & Beedle, M. (2001). Agile Software Development with Scrum (1st ed.). Upper Saddle River, NJ, USA: Prentice Hall PTR.
25. Sharp, H. (2008). Customer collaboration in distributed agile teams. Proceedings of Distributed Participatory Design Workshop. Florence.
26. Sharp, H., Robinson, H., & Petre, M. (2009). The role of physical artefacts in agile software development: Two complementary perspectives. Interacting with Computers , 108-106.
27. Stelzer, D., & Mellis, W. (1998). Success Factors of Organizational Change in Software Process Improvement. SOFTWARE PROCESS IMPROVEMENT AND PRACTICE , 227-250.

28. Turner, R. (2003). People factors in software management: lessons from comparing agile and plan-driven methods. J Def. Software Engineering , 4-8.
29. Voas, J. M., & Whittaker, J. A. (2002). 50 years of software: key principles for quality. IT Professional , 28-35.
30. Wauter, R. (2009, 01 22). Sad Day For Microsoft: 5,000 Laid Off, Earnings And Revenues Down. Retrieved 04 30, 2009, from TechCrunch: http://www.techcrunch.com/2009/01/22/sad-day-for-microsoft-5000-laid-off-earnings-and-revenues-down/