International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 02 Issue: 08 | Nov-2015 www.irjet.net p-ISSN: 2395-0072

Literature Survey on Agile Information Systems Development

Surbhi R. Khare[1], Ritesh Shrivastava[2]

[1] PIET, Department of Computer Technology, Nagpur, INDIA,
[2.] ACET, Department of CSE, Nagpur, INDIA

Abstract— The principles of agile information systems development (ISD) have become an area of great interest for practice as well as research. So the goal of this literature survey is to validate, update and extend previous reviews in terms of the general state of research on agile ISD. The importance of theory is highlighted by evaluating the theoretical foundations and contributions of former studies besides including categories such as the employed research methods and data collection techniques. Since agile ISD is rooted in the IS as well as software engineering discipline, important outlets of both disciplines are included in the search process. The findings show that quantitative studies and the theoretical underpinnings of agile ISD are lacking. Extreme Programming is still the most researched agile ISD method, and more efforts on Scrum are needed. In addition, multiple research gaps that need further research attention are also identified.

Key Words: *Agile Process, Agile Method, Agile Practice, Agile Principles*

----------***----------

1.Introduction

Agile software development is a group of software development methods in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages rapid and flexible response to change.

Agile methods for software and information systems development (ISD) such as Scrum [1] or Extreme Programming (XP) [2] are very popular in industry. Those methods complement the iterative approach to ISD [1], [3], [4] and have been suggested as a way to react quickly to changing requirements by emphasizing small release cycles and through continuous integration of the customer [5] [6] [7]. In contrast to traditional methods, flexibility and autonomy is considered important, the overall project is not planned and scheduled upfront, and the development process is split in small iterations, while encouraging constant feedback of the customer [5], [6]. Consequently, agile ISD methods appear to incorporate many lessons learned about ISD during the past [7], [8].

Following the guidelines of Webster & Watson [9] and Peterson et al. [10], this literature review provides insights into the general state of research on agile ISD in terms of research approaches, methods, data collection techniques, and focus of the studies. The state of theory in the field of agile ISD is also evaluated by looking at theoretical contributions, employed theories and definitions of agility. The goal of this review is to identify research areas that deserve future attention of the research community. Consequently, the following two-part research question is investigated: What is the state of research on agile lSD, and in consequence, what are the implications for future studies?

The remainder of this paper is structured as follows. First, related literature reviews on agile ISD are briefly discussed, followed by the agile principles including details on the agile development methods. As a next step, the results of the literature review are presented. In the last section, the findings are summarized and implications for future research are presented, including several research gaps that entail opportunities for future work.

2 RELATED WORK

The first reviews were provided by Abrahamsson et al. [11], Cohen et al. [12], and Erickson et al. [13]. These reviews focus on the employed agile practices and methods in industry. For example, Erickson et al. [13] review the state of research on XP and agile modeling. The study finds that most empirical research is concerned with the XP practice pair programming, and other practices are neglected. A more holistic review of agile ISD is provided by Dyba & Dingseyr [14] who investigate studies up to and including 2005. The authors classify research on agile ISD in four main themes: introduction and adoption, human and social factors, perceptions on agile methods, and comparative studies. They find that there is a need for more rigorous, high quality empirical studies. Another finding is that there is a need for conducting research based on other methods besides XP because empirical evidence on popular methods such as Serum is missing.

Most recently, Dingsoyr et al. [15] provide an overview of the theoretical perspectives that are employed by research on agile lSD, but as the authors state themselves, the search results are limited because only the topic of studies were searched, and the search strings were based on a previously defined keyword list of twenty theoretical perspectives, including lightweight theoretical perspectives such as knowledge management and personality. The study concludes in a call for a more theory-based research approach in the field of agile ISD.

3. AGILE PRINCIPLES

The various agile principles are as follows.
The Agile Manifesto is based on 12 principles:

1 Customer satisfaction by rapid delivery of useful software
2 Welcome changing requirements, even late in development
3 Working software is delivered frequently (weeks rather than months)
4 Close, daily cooperation between business people and developers
5 Projects are built around motivated individuals, who should be trusted
6 Face-to-face conversation is the best form of communication (co-location)
7 Working software is the principal measure of progress
8 Sustainable development, able to maintain a constant pace
9 Continuous attention to technical excellence and good design.
10 Simplicity—the art of maximizing the amount of work done—is essential Self-organizing teams.
11 Regular adaptation to changing circumstance.
12 Here are many specific agile development methods. Most promote development, team work and process adaptability throughout the life-cycle of the project.

3.1 Iterative, incremental and evolutionary

Most agile development methods break tasks into small increments with minimal planning and do not directly involve long-term planning. Iterations are short time frames (timeboxes) that typically last from one to four weeks. Each iteration involves a cross-functional team working in all functions: planning, requirements analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is demonstrated to stakeholders. This minimizes overall risk and allows the project to adapt to changes quickly. An iteration might not add enough functionality to warrant a market release, but the goal is to have an available release (with minimal bugs) at the end of each iteration.[12] Multiple iterations might be required to release a product or new features.

3.2 Efficient and face-to-face communication

No matter what development disciplines are required, each agile team contains a customer representative, e.g. *product owner* in scrum. This person is appointed by stakeholders to act on their behalf[13] and makes a personal commitment to being available for developers to answer mid-iteration questions. At the end of each iteration, stakeholders and the customer representative review progress and re-evaluate priorities with a view to optimizing the return on investment (ROI) and ensuring alignment with customer needs and company goals.

In agile software development, an *information radiator* is a (normally large) physical display located prominently in an office, where passers-by can see it. It presents an up-to-date summary of the status of a software project or other product.[14][15] The name was coined by Alistair Cockburn, and described in his 2002 book *Agile Software Development*.[15] A build light indicator may be used to inform a team about the current status of their project.

3.3 Very short feedback loop and adaptation cycle

A common characteristic of agile development are daily status meetings or "stand-ups", e.g. *daily scrum (meeting)*. In a brief session, team members report to each other what they did the previous day, what they intend to do today, and what their roadblocks are.

3.4 Quality focus

Specific tools and techniques, such as continuous integration, automated unit testing, pair programming, test-driven development, design patterns, domain-driven design, code refactoring and other techniques are often used to improve quality and enhance project agility.

4.AGILE METHODS

Well-known agile software development methods and/or process frameworks include:

4.1 Adaptive software development (ASD):

Adaptive software development (ASD) is a software development process that grew out of rapid application development work by Jim Highsmith and Sam Bayer. It embodies the principle that continuous adaptation of the process to the work at hand is the normal state of affairs.

Adaptive software development replaces the traditional waterfall cycle with a repeating series of *speculate*, *collaborate*, and *learn* cycles. This dynamic cycle provides for continuous learning and adaptation to the emergent state of the project. The characteristics of an ASD life cycle are that it is mission focused, feature based, iterative, timeboxed, risk driven, and change tolerant.

4.2 Agile modeling :

Agile modeling (AM) is a methodology for modeling and documenting software systems based on best practices. It is a collection of values and principles, that can be applied on an (agile) software development project. This methodology is more flexible than traditional modeling methods, making it a better fit in a fast changing environment.[1] It is part of the Agile software development tool kit.

Agile modeling is a supplement to other agile methodologies such as Scrum, extreme programming (XP), and Rational Unified Process (RUP). It is explicitly included as part of the disciplined agile delivery (DAD) framework. As per 2011 stats, agile modeling accounted for 1% of all agile software development.

4.3 Agile Unified Process (AUP):

Agile Unified Process (AUP) is a simplified version of the Rational Unified Process (RUP) developed by Scott Ambler.[1] It describes a simple, easy to understand approach to developing business application software using agile techniques and concepts yet still remaining true to the RUP. The AUP applies agile techniques including test-driven development (TDD), Agile Modeling (AM), agile change management, and database refactoring to improve productivity.

4.4 Dynamic systems development method (DSDM) : Dynamic systems development method (DSDM) is an project delivery framework, primarily used as a agile.[1][2] First released in 1994, DSDM originally sought to provide some discipline to the rapid application development (RAD) method.[3] In 2007 DSDM became a generic approach to project management and solution delivery. DSDM is an iterative and incremental approach that embraces principles of Agile development, including continuous user/customer involvement.

DSDM fixes cost, quality and time at the outset and uses the MoSCoW prioritisation of scope into *musts*, *shoulds*, *coulds* and *won't haves* to adjust the project deliverable to meet the stated time constraint. DSDM is one of a number of Agile methods for developing software and non-IT solutions, and it forms a part of the Agile Alliance.

4.5 Extreme programming (XP):

Extreme programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development,[1][2][3] it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted.

4.6 Feature-driven development (FDD):

Feature-driven development (**FDD**) is an iterative and incremental software development process. It is one of a number of lightweight or Agile methods for developing software. FDD blends a number of industry-recognized best practices into a cohesive whole. These practices are all driven from a client-valued functionality (feature) perspective. Its main purpose is to deliver tangible, working software repeatedly in a timely manner.

4.7 Scrum :

Scrum is an iterative and incremental agile software development framework for managing product development. It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal", challenges assumptions of the "traditional, sequential approach" to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines in the project.

5 CONCLUSION

In this study, a systematic, structured literature review in the field of agile ISD was conducted. The results show that the state of research on agile ISD is still nascent because there is an imbalance in terms of the employed research methods towards interview-based case studies. Those qualitative research designs are essential for providing first evidence on important factors and relationships, but confirmatory studies testing the qualitative findings are lacking. More studies are needed that are based on quantitative approaches such as field studies or experiments in order to ensure that the qualitative findings are generalizable. Furthermore, this review exposed promising research areas by presenting a systematic map on the focus of the studies and the employed research methods. Some of these areas, for example the communication patterns of agile teams, are highly important for the successful implementation of agile practices [6] and first qualitative findings exist , but research remains scarce. The same proposition holds for other research areas such as agile vs. lean, hybrid approaches and organizational culture. In consequence, we need more studies that address the following research gaps:

1. What are the implications of agile ISD on the coordination, collaboration and communication *mechanisms within agile teams?*
2. *How are agile ISD and lean software development related?*
3. *What is the impact of agile practices on the organizational culture?*
4. *How can agile methods and traditional, plan based methods be combined?*
5. *What are the implications of agile ISD on release scheduling and requirements engineering?*
6. *What are the success factors underlying agile ISD?*

Despite the popularity of Scrum in industry, most researched is based on XP, more specifically on the pair programming, unit testing and refactoring practices. One possibled from the original submission for page layout reasons. This includes the possibility that some in-line equations will be made display equations to create better flow in a paragraph. If display equations do not fit in the two-column format, they will also be reformatted. Authors are strongly encouraged to ensure that equations fit in the given column width. reason for this emphasis on XP is that studies on pair programming may be set up inexpensively in an academic setting with small teams of students. In terms of unit testing and refactoring, many studies propose tools that may support those practices. Future research should focus on other XP practices such as collective code standards and on-site customer. Furthermore, more research is needed that provides theoretically grounded guidance for industry on the adoption, adaption and success factors of Scrum.

6 REFERENCES

[1] K. Schwaber, and M. Beedle, Agile Software Development with Scrum, Prentice Hall, Upper Saddle River, NJ, USA, 2002.
[2] K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley, Boston, MA, USA, 1999.
[3] M. Poppendieck, and T. Poppendieck, Lean Software Development: An Agile Toolkit, Addison-Wesley Longman, 1st ed., Amsterdam, 2003.
[4] J. Martin, Rapid application development, Macmillan Publishing Co., Inc, New York, NY, USA, 1991.
[5] L. Cao, K. Mohan, X. Peng, and B. Ramesh, "A framework for adapting agile development methodologies",European Journal of Information Systems, 18(4), 2009, pp.332-343.
[6] R. Vidgen, and X. Wang, "Coevolving Systems and the Organization of Agile Software Development", Information Systems Research, 20(3), 2009, pp. 355-376.

[7] J. Highsmith, and A. Cockburn, "Agile SoftwareDevelopment: The Business of Innovation", IEEE Computer, 34(9), 2001, pp. 120-127.
[8] A. Cockburn, and J. Highsmith, "Agile Software Development: The People Factor", IEEE Computer, 34(11), 2001, pp. 131-133.
[9] J. Webster, and R.T. Watson, "Analyzing the past to prepare for the future: Writing a literature review", MIS Quarterly, 26(2), 2002, pp. 13-23.
[10] K. Peterson, R. Feldt, S. Mujtaba, and M. Mattsson,"Systematic mapping studies in software engineering", in: International Conference on Evaluation and Assessment in Software Engineering, 2008, pp. 68-77.
[11] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, Agile software development methods: review and analysis, VTT Technical Report, 2002.
[12] D. Cohen, M. Lindvall, and P. Costa, "An introduction to agile methods", in (Zelkowitz, M.V.): Advances in Computers, Elsevier Ltd., .
[13] J. Erickson, K. Lyytinen, and S. Keng, "Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research", Journal of Database Management, 16(4), 2005, pp. 88-100.
[14] T. Dyba, and T. Dingsoyr, "Empirical studies of agile software development: A systematic review", Information and Software Technology, 50(9-10), 2008, pp. 833-859.
[15] T. Dingsoyr, S. Nerur, V. Balijepally, and N.B. Moe,"A decade of agile methodologies: Towards explaining agile software development", Journal of Systems and Software, 85(6), 2012, pp. 1213-1221.
[16] D. Batra, D.E. Vandermeer, and K. Dutta, "Extending Agile Principles to Larger, Dynamic Software Projects: A Theoretical Assessment", Journal of Database Management, 22(4), 2011, pp. 73-92.
[17] G. Lee, and W. Xia, "Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data", MIS Quarterly, 34(1), 2010, pp. 87-114.
[18] T.D. Hellmann, A. Hosseini-Khayat, and F. Maurer, "Test-Driven Development of Graphical User Interfaces: A Pilot Evaluation", in (Sillitti, A., Hazzan, O., Bache, E., and Albaladejo, X.): Agile Processes in Software Engineering and Extreme Programming, XP 2011, Springer, Berlin Heidelberg, 2011, pp. 223-237.