COMPARISON OF AGILE PROCESS MODELS TO CONCLUDE THE EFFECTIVENESS FOR INDUSTRIAL SOFTWARE PROJECTS

M. Rizwan Jameel Qureshi, Fuad Bajaber
Faculty of Computing and Information Technology, King Abdulaziz University,
Jeddah, Kingdom of Saudi Arabia
rmuhammd@kau.edu.sa, fbajaber@kau.edu.sa
Cell # (+966-536474921)

ABSTRACT: The aim of agile principles is to develop small size software projects. There is no assistance how to tailor agile methodologies for the development of medium size and complicated software. There are several agile models proposed in the history of software engineering such as XP, Feature Driven Development (FDD), Adaptive Software Development (ASD) and Dynamic System Development Model (DSDM). It is extremely hard choice for a software company to select a suitable agile methodology to tailor it for in house development. There are several case studies reported about the experiences of agile software development by several researchers. There is still requirement to study and compare agile software development process models. The same is accomplished in this research by comparing agile models with their pros and cons. Further, XP and Scrum are compared by conducting two controlled case studies due to their widespread usage to estimate quality that which of the model is better than the other. Scrum is found to be more effective than XP by showing high quality.

Key words: SDLC, Process Models, Agile Principles, Quality Attributes

1. INTRODUCTION

Software engineering is a paradigm that is composed of methodologies, techniques and tools [1,2]. Methodologies are mainly divided into structured and object oriented. Object oriented is widely practiced in software companies from last several years due to its profound benefits over structured methodologies like reusability, time saving and less cost [3,4]. The most widely practiced object oriented methodologies are Rational Unified Process (RUP) and Agile. Agile methodologies are selected for this paper due to its popularity and active area of research [5,6]. Several controlled and industrial case studies are reported in the literature about the successful implementations of agile methodologies [2,6].

Agile methodologies focus mainly on agility to develop software timely and within budget. Agility is defined as to adapt changes during software development as per the needs to achieve success without compromising quality. Agile alliance defined twelve golden principles in 2001 to achieve successful software development [2,7].

- Customer satisfaction by providing increments at continues intervals of software development.
- Deliver the first increment within two to three weeks and complete software within two to three months.
- Continues interaction of customer and agile throughout software development.
- Physical meetings between team and customer.
- Customer has the privilege to provide new requirements and even change requirements at any stage of system development.
- Faith and regard among agile team.
- Measure the pace of the project at consistent intervals during the software development.
- Good design always results into high quality.
- Self-disciplined persons always produce high quality architecture and design.
- Adaptation of team and process is required as per the conditions of software development.
- Agile team must follows keep it simple (KIS) principle to design and develop software.
- Dedicated team must be allocated to develop agile software.

Twelve golden principles proposed by the agile alliance are difficult to meet in the current global software development environment following agile methodologies. Agile methodologies do not support distributed development teams, subcontracting, reusable component based development (CBD), sizeable development teams, safety critical projects, development of large and complicated software [8,9] and strong documentation [10,11].

The remainder of this paper is organized as follows: Section 2 illustrates the related work. Section 3 covers the comparison of popular agile models. Section 4 provides the validation using two controlled case studies. Section 5 illustrates the discussion to conclude the results.

2. RELATED WORK

Several case studies are documented to support the agile software development from last several years [6][12-14]. The case studies are conducted to complete projects between two to three months. Both qualitative and quantitative techniques are used to provide results. The results show that there must be a process to check the performance of team and procedures of software development to increase efficiency of both. It is also reported that agile practices decrease defect rate and increase productivity of software development. There is no evidence in any of the case studies that how to tailor agile methodologies for average and complex software projects.

Another case study is reported to increase awareness of software developers about agile methodologies [15]. It is a Ph.D. dissertation and does not provide experimental results to manage the problems of software industry.

The most popular agile methodologies are XP, Adaptive Software Development (ASD), Dynamic System Development Method (DSDM), Change-Oriented Life Cycle and Feature Driven Development (FDD) [16]. FDD

Table 1-The Comparison of Agile Methodologies

Main Agile Methodologies	Main Limitations
XP [18-21]	Main limitations of XP are poor documentation. It is also inappropriate for distributed teams, reuse and subcontracting and development of average and complex software. The success of XP mainly depends on the support of its stakeholders.
Scrum [22-24]	Scrum is mainly a management framework than a methodology. It doesn't support for the development of large software and sizeable teams. Scrum also does not assist that how to complete an iteration in one month.
DSDM [2][25]	DSDM does not handle the engineering of average and complex projects. There is no support for the management of sizeable teams. There is a provision to compose DSDM with XP methodology but at the expense to manage limitations of XP that are infused into DSDM.
Crystal Family [26]	Geographically distributed teams are not supported by Crystal family of models. More work is needed to measure usefulness of Crystal family of methodologies for the engineering of all types of projects.
FDD [27]	FDD is the most recently proposed methodology as compared to other agile methodologies. More validation is required to be widely practiced in software industry.

methodology is supported by reporting a case study of fifteen months project. The core objective of the case study is to show the role of agile team on a successful software project. The case study further reports how to decrease the pressure and danger of agile methodologies on agile teams and projects. It is also discussed that it is hard to manage large projects with sizeable teams using agile methodologies. It is also recommended that maximum team size is nine persons following agile methodologies. The case study does not report enough information that how FDD methodology is implemented over 15 months project to achieve success [16].

It is reported that transition, from traditional to agile methodologies, has a strong impact to entire software company groups including developers, managers and admin [17]. Following are the results of case study [17].

- Agile projects may fail in the presence of such team members who are too anxious or against to implement rapid changes.
- Agile projects are also in danger if managers do not communicate with their teams on daily basis to fix their problems.
- Agile project also fails if transition from traditional to agile methodology is not steady.
- Agile teams may fail in case of switching from collocated teams to distributed teams without training.
- The skills of a successful team are expertise, analytical, management, teamwork, goal centric, faith, regard and self-discipline.
- It is necessary to organize agile team in such a way that programmers and testers are sitting together to improve agility and fix the bugs immediately as reported.
- It is required to consider as a condition using agile methodologies that top management of software company does not commit unrealistic deadlines to a customer without taking confidence to its teams to achieve success.

The results are just guidelines based on the experiences of selected case studies and these are tested in other settings to generalize the results [17]. Changes are accepted throughout the software development using agile methodologies. There is always uncertainty for agile or non agile project that it will be successful or not. The uncertainty even further increase if team is using agile methodology first time. The chances of success highly depend on the fact that how much team adapts to changes facing during the agile project.

Main limitations of XP are poor documentation. It is also inappropriate for distributed teams, reuse and subcontracting and development of average and complex software. The success of XP mainly depends on the support of its stakeholders [18-21]. Scrum is mainly a management framework than a methodology. It doesn't support for the development of large software and sizeable teams. Scrum also does not assist that how to complete an iteration in one month [22-24].

DSDM does not handle the engineering of average and complex projects. There is no support for the management of sizeable teams. There is a provision to compose DSDM with XP methodology, but at the expense to manage limitations of XP that are infused into DSDM [2,25]. Geographically distributed teams are not supported by Crystal family of models. More work is needed to measure usefulness of Crystal family of methodologies for the engineering of all types of projects [26]. FDD is the most recently proposed methodology as compared to other agile methodologies. More validation is required to be widely practiced in software industry [27].

A customized XP methodology is proposed to implement a problem-solving information system [28]. It is an integrated methodology that is a combination of XP and tailored Waterfall methodologies. The aim to propose the customized methodology to develop knowledge based systems following agile principles and practices to decrease time and cost of development. It is too early to predict the effectiveness of customized XP methodology because the problem-solving system is under construction.

Two case studies are presented using adapted XP model [29-31]. A case study is performed in IBM for twelve months time. Second case study is performed in Sabre Airline Solutions for ninety days. The adapted XP model is named as extreme programming evaluation framework (XP-EF). A feedback loop is introduced in XP-EF to estimate the performance of agile team and practices. More work is required that XP-EF meets the XP adherence metrics. Both case studies cannot be used as a standard to apply XP-EF in other settings because of following reasons.

- The teams in both case studies were well experienced to apply agile methodologies.

- There was complete management assistance to conduct both case studies [29-31].

3. COMPARISON OF AGILE METHODOLOGIES

Plan, Design, Code and Test are four phases of XP methodology [2]. It is suggested using the core principles and lessons learned from its ancestor methodologies [18]. Following are the two main concepts derived from the previous methodologies.

- A project is always planned based on the user stories those are depending on the uses cases like the Rational Unified Process (RUP) [2,18].
- XP is incremental like its ancestor evolutionary methodologies [2,18].

The main advantages of XP methodology are timely delivery, economical, refactoring and appropriateness for the development of small size software using small size teams [2,18]. Improve the architecture and program using Refactoring method throughout the SDLC. The main limitation of XP methodology is inappropriate for the development of average and complex software due to poor documentation. It does not support to reuse due to fast delivery cycle. Global software development and subcontracting are also not supported using XP [8]. XP has good engineering practices, but lacks in management practices. The successful XP stories show that it requires full assistance from its stakeholders [8,18].

The term 'Scrum' is used in Rugby game [2]. The Scrum methodology is suggested in the early nineteen nineties [2]. Scrum is more like a framework than a methodology. It is strong in management practices. Product backlog, sprint backlog, effort estimation, sprint meetings, daily meetings and burndown chart are the main activities of Scrum [2]. Scrum master, Product owner and Scrum teams are the main roles using Scrum. Scrum supports to implement small scale using five to seven team members [23]. The main limitation of Scrum is inappropriate for the development of average and complex software projects [22]. Scrum does not support to large size teams. There is no recommendation that how to apply Scrum to complete a sprint using 30 days release cycle [32].

DSDM methodology is introduced in nineteen ninety four [25]. DSDM is similar to Scrum with respect to team size. It works with small teams. DSDM has shown its effectiveness to develop business applications. DSDM is ineffective to develop scientific or engineering applications [25]. It is highly effective to develop small size projects. DSDM is inappropriate for the development of average and complex projects. The principles and practices of DSDM are monitored, controlled and improved by a consortium [32]. DSDM has the privilege to enrich its benefits by composing with other agile methodologies like XP. The limitations of XP are also inherited into DSDM [2].

Highsmith and Cockburn proposed Crystal methodologies [21]. Crystal methodologies are grouped into three types, i.e., Clear, Orange and Orange Web [26]. Business applications, with less than 6 team members, are developed using Crystal Clear. Engineering and scientific applications are developed using Crystal Orange. The recommended team size is 10 to 40 members to use Crystal Orange methodology [26]. There is no application/case study reported using Crystal Orange Web in the current literature [26]. It is proposed to develop parallel applications [26]. Crystal methodologies are proposed with the intension to provide a selection to the software companies that they can pick an appropriate methodology as per the type of project [26]. The main limitation of Crystal methodologies is that these methodologies are still at the stages of not more than proposals. Empirical validations are required to test Crystal methodologies to conclude solid results (using qualitative and quantitative techniques) before these methodologies can be applied in industrial settings [32]. There is no support for global software development using Crystal methodologies because of recommended closed physical interaction among team members [32].

Table 2-The Data of Two Controlled Case Studies

Items	XP	Scrum
Type of Information System	Library	Payroll
Team size	6	6
Calendar Time (weeks)	5	5
Releases	4	4
Total Tasks defined	96	82
Total work effort (h)	1260	1000
Team Productivity	41	46
Post release defects	20	16
Customer Satisfaction	82%	85%

Five procedures are proposed in FDD instead of SDLC phase [27]. An empirical study is conducted in the nineteen nineties by applying FDD on an enterprise planning system [27]. The results recommend that FDD can also be applied to maintain software [27]. FDD inherits the common limitations those are found in other agile methodologies [32]. More controlled case studies are required to test FDD methodology to generalize the results before it can be applied in software industry for the development of commercial software. It is discussed in [32], there is no notably work found about the successful implementations of FDD in software industry.

A comparison of the main limitations of commonly practiced agile methodologies is shown in Table 1. The existing literature shows that XP and Scrum are the most commonly used agile methodologies [2][18]. To show the effectiveness of agile models, XP and Scrum are selected in this research to conclude the results by conducting two controlled case studies.

4. VALIDATION

Two controlled case studies are conducted, i.e., one for Scrum model and second for the XP model. Both case studies are conducted in the premises of COMSATS Institute of Information Technology, Lahore, Pakistan. A team of six members are selected to conduct both case studies to generalize the results. The details of both case studies are provided in the sub sections 4.1 and 4.2. Table 2 shows the results of XP and Scrum case studies.

4.1 Scrum Case Study

The basic purpose of conducting the case study is to build a system following the Scrum methodology. The duration of case study was five weeks. The case study is used to develop a financial system. The principles and practices of Scrum methodology are followed to complete the first four iterations of financial system to infer the results. The team has already completed the term projects during Software Engineering I and II courses, but it is the first experience of team to implement the Scrum. Therefore, a training program is arranged at the start of case study to literate the team about the principles and practices of Scrum. The practices covered during training are sprint zero, product backlog, sprint backlog, sprint planning meeting, daily scrum meeting, sprint review meeting, and sprint retrospective. The Scrum team is composed of six members i.e., Scrum master, 3 designers/developers and 2 testers. Scrum master is responsible to handle team. Scrum master acts the role of product owner as well. Rational Rose, Net Beans, My SQL, J-Unit, and Ireport tools are applied during the case study.

4.2 XP Case Study

The length of XP case study is five weeks. The case study is used to develop an online Library Management System (LMS). The results are accomplished using the first four iterations of LMS. A training program is organized for the team to get familiar with XP principles and practices. The team already has the experience to implement agile development using Scrum case study in terms of procedures, roles and artifacts. The main practices covered during training are keep it simple (design, code and documentation), worked in pairs, automated testing, integrate immediately after completion, code following standards and on-site customer.

5. DISCUSSION OF THE RESULTS

Table 2 shows that a team of six members is selected to develop Payroll and Library systems of an educational institute using Scrum and XP subsequently. The results are concluded on the first four iterations of two case studies. The results are shown on the average of first four iterations. Calendar time is five weeks for the accomplishment of four releases. The results show that ninety six tasks are allocated using the XP as compared to eighty two tasks using the Scrum. XP consumed twelve sixty hours whereas Scrum utilized one thousand hours work effort. Productivity is calculated using user stories per person month in both case studies. XP shows low productivity as compared to Scrum i.e., 41 vs. 46. XP shows twenty post release defects as compared to sixteen using the Scrum. Scrum shows high customer satisfaction as compared to XP i.e., 85% vs. 82%. A survey is conducted from the customer after each release to calculate satisfaction.

The results indicate that Scrum has high quality as compared to XP due to it's a better management practices. XP completes more number of tasks in the same calendar time but at the expense of high work effort using the same team size. Thus Scrum is a better model than XP inferring from the results of two case studies.

6. CONCLUSION

There are several agile models proposed from last several years. It is a difficult choice for a software company to select a suitable model following agile principles. The aim of this research is to compare agile model with their pros and cons to facilitate project managers. XP and Scrum are the most widely practiced agile models. Two controlled case studies are conducted, i.e., one for XP and second for Scrum. The objective is to compare the strengths and weaknesses of XP and Scrum. Scrum shows significant performance over XP by showing lesser number of defects and more customer satisfaction.

REFERENCES

[1] Sebastian, T. (2004). The Many Dimensions of the Software Process. Crossroads ACM Press, Vol. 6(4), 22-26.

[2] Pressman, R. S. (2011). Software Engineering. New York, McGraw Hill.

[3] Bruynooghe, R. F. Greenwood, R.M. Robertson, I. Sa, J. Snowdon, R.A. and Warboys, B.C. (1994). PADM: Towards a Total Process Modelling System. Software Process Modelling and Technology, Research Studies Press, 293-334.

[4] Greenwood, M. Warboys, B.C. and Sa, J. (1996). Cooperating evolving components: a rigorous approach to evolving large software systems. Proc. 18th Int. Conf. Software Engineering, Berlin, Germany, 428-437.

[5] Schmietendorf, A. Dimitrov, E. and Dumke, R. R. (2002). Process Models for the software development and performance engineering tasks, Proc. 3rd Int. workshop on Software and performance, Rome, Italy, 211-218.

[6] Outi, S. and Pekka, A. (2004). Empirical Evaluation of Agile Software Development: The Controlled Case Study Approach. Proc. 5th Int. Conf. Product Focused Software Process Improvement, 408-423.

[7] Agile Alliance (2001). http://agilemanifesto.org/principles.html, Visited July 23, 2016.

[8] Turk, D. France, R. and Rumpe, B. (2002). Limitations of Agile Software Processes. Proc. 3rd Int. Conf. eXtreme Programming and Agile Processes in Software.

[9] Highsmith, J. and Cockburn, A. (2001). Agile Software Development: The People Factor. Computer, Vol. 34(11), 131-133.

[10] McCarey, F. (2005). Agile software reuse recommender. Proc. 27th Int. Conf. Software engineering, St. Louis, MO, USA, 652-652.

[11] VIT publications (2002). http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf, Visited July 23, 2016.

[12] Outi, S. (2004). Improving Software Process in Agile Software Development Projects: Results from Two XP Case Studies. Proc. 30th EUROMICRO Conference, France, 310-317.

[13] Outi, S. Kari, K. Pekka, K. Jani, L. Sanna, S., and Pekka, A. (2004). Self-Adaptability of Agile Software Processes: A Case Study on Post-iteration Workshops. Proc. 5th Int. Conf. Extreme Programming and Agile

Processes in Software Engineering, Germany, 184-193.
[14] Outi, S. Minna, P. Jari, S. (2005). Deploying Agile Practices in Organizations: A Case Study. Proc. European Conference on Software Process Improvement (EuroSPI 2005), Hungry, 16-27.
[15] Cao, L. (2004). Modeling Dynamics of Agile Software Development. Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, Vancouver, BC, CANADA, pp. 46-47.
[16] Highsmith, J. and Cockburn, A. (2001). Agile Software Development: The Business of Innovation. Computer 34(9), 120-122.
[17] Cohan, M. and Ford, D. (2003). Introducing an Agile Process to an Organization. Computer, Vol. 36(6), 74-78.
[18] Beck, K. (1999). Embracing Change with Extreme Programming. IEEE Computer, Vol. 32(10), 70-77.
[19] Beck, K. (2000). Extreme Programming Explained: Embrace Change. USA, Addison Wesley.
[20] Beck, K. (2003). Test-Driven Development By Example. USA, Addison Wesley.
[21] Beck, K. and Andres, C. (2004). Extreme Programming Explained: Embrace Change. Boston, Addison Wesley.
[22] Schwaber, K. (1995). Scrum Development Process. Proc. Of the OOPSLA'95 Workshop on Business Object Design and Implementation. Springer-Verlag, 117-134.
[23] Schwaber, K. and Beedle, M. (2002). Agile Software Development with Scrum. USA, Prentice Hall.
[24] Schwaber, K. (2004). Agile Project Management with Scrum. USA, Microsoft Press.
[25] Stapleton, J. and Peter, C. (1997). DSDM Dynamic Systems Development Method: The Method in Practice, USA, Addison Wesley.
[26] Cockburn, A. (2005). Crystal Clear: A Human-Powered Methodology for Small Teams. USA, Addison Wesley.
[27] Palmer, S. R. and Felsing, J. M. (2002). A Practical Guide to Feature-Driven Development. USA, Prentice Hall.
[28] Baumeister, J., B. Puppe , F. and Dietmar, S. (2004). An Agile Process for Developing Diagnostic Knowledge Systems. KI Journal, Vol. 18(3), 12-16.
[29] Williams, L., Kerbs, W., Layman, Anton, A. and Abrahamsson, P. (2004). Toward a Framework for Evaluating Extreme Programming. Proc. 8th Int. Conf. Empirical Assessment in Software Engineering, Edinburgh, Scotland, 11-20.
[30] Layman, L., Williams, L. and Cunningham, L. (2004). Exploring Extreme Programming in Context: An Industrial Case Study. Proc. 2nd Agile Development Conf., Salt Lake City, UT, 32-41.
[31] Layman, L., Williams, L. and Cunningham, L. (2006). Motivations and Measurements in an Agile Case Study. Journal of Systems Architecture, Vol. 52(11), 654-667.
[32] Abrahamsson, P. Juhani, W. Mikko, T. S. and Jussi, R. (2003). New Directions on Agile Methods: A Comparative Analysis. Proc. 25th Int. Conf. Software Engineering, Portland, Oregon, 244-254.