Sustainable Software Development: A Comparison of Tailored Agile Processes

Sophia McNamarah, M.Sc[1], Sherrene Bogle[2], PhD, Richard Pyne, PhD[3]
[1]University of Technology, Jamaica, smcnamarah@utech.edu.jm
[2]Humboldt State University, sherrene.bogle@humboldt.edu
[3]Sheridan College, Canada, richard.pyne@sheridancollege.ca

***Abstract*– Agile methods have the potential for cost effective and efficient software development thereby facilitating economic sustainability. A well-defined process is associated with a better quality software product and a sustainable production flow. However, agile processes have their shortcomings such as inadequately detailed user stories and lack of overall (architectural) design. Such shortcomings can affect the effectiveness of the agile process. Using principles of process tailoring, three agile processes are tailored by adding steps that address their weaknesses. The tailored processes once derived were implemented and tested. The results of the controlled case study show that the altered extreme programming resulted in better cost effectiveness and may be a useful angle for enhancing economic sustainability.**

***Keywords* -- Agile development, Process tailoring, Software development.**

I. Introduction

Over half of the world's population live in cities and it is expected that the number of city dwellers will increase to as much as 86% of the population in some countries over the next few decades [1]. Cities provide jobs and economic development, however growing cities threaten sustainability because growing urban population mean greater use of energy, land and other resources [1]. Sustainability, (economical, ecological or social), refers to the principle of meeting current needs without interfering with the ability to meet future needs [2]. Economic sustainability can be enhances if cities can achieve wealth through improved productivity and competitiveness [3]. Production and manufacturing processes such as agile development and lean manufacturing can help to improve productivity and reduce waste [4].

Over the last decade agile software development methodologies have become more popular as more companies have adopted agile practices. Agile methods have the potential for cost effectiveness and efficiency since they are lightweight and produce tangible output at a faster rate [5]. This improvement in cost effectiveness and efficiency can help cities in terms of economic sustainability.

Agile methodologies consist of a range of practices, which are responsive to change in the development process. They are based on four underlying principles viz. individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; and responding to change over following a plan [6] [7].

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2019.1.1.458
ISBN: 978-0-9993443-6-1 ISSN: 2414-6390

II. Related Work

Agile software development methods are characterized by accommodating changes in the requirements throughout the development process [8]. They are typically evolutionary and/or iterative; they permit and accommodate requirements changes during the development; they select, adjust and develop functionality and features according to fixed time and resources; and they have continuous testing and integration [9]. Agile software development processes include Extreme Programming (XP), Scrum, Crystal, Feature Drive Development (FDD), Lean Development and Dynamic Systems Development method (DSDM) among others. The fundamental similarity among agile methods is that they were designed to accommodate change. In addition they tend to be evolutionary and incremental rather than purely sequential and they all have less documentation than traditional methods. Agile methods are better suited for small teams working in the same location [10] [11] [12]. This actually enhances economic sustainability since data show that in some of the leading software developing cities small businesses with small teams make up the vast majority of software development companies [13]. Agile processes also have several releases of the product (which usually evolves over time), continuous integration and active customer involvement usually throughout the entire development process.

A. Agile benefits and shortcomings

In addition to their increasing popularity agile methods provide a range of benefits. Agile methods enable ongoing day-to-day project tracking and monitoring, real-time updates, rapid delivery of products, greater interactions and collaborations between stakeholders, improved communications, knowledge sharing and quick feedback [14] [15] [16]. Agile processes can provide a competitive advantage for small companies since their small releases provide focus and faster delivery. Small companies need mechanisms for cost saving and for prudent use of resources.

Yet despite their range of benefits agile processes have their own shortcomings. These shortcomings sometimes arise as a result of the organization of the agile process. One such shortcoming is the use of user stories to capture requirements for the software. User stories tend to lack sufficient details for requirements [17] [18] [19]. Another such shortcoming is the

use of little or no documentation, which can be problematic, for example it does not permit team members to gain a broad overview of the system from documentation, it makes things challenging when new persons join the agile team, it can cause defects when the software is to evolve or be maintained, it can lead to difficulty tracing changes and difficulty tracing design decisions [20] [21] [22] [23].

Another major shortcoming is the little or no architectural design [18] [24 [25] This too prevents the team from having an overview of the system and an understanding of how the components and functionalities fit together and their dependencies [18]. Barbar [25] stated that many agile practitioners view the upfront evaluation and design of the software architecture as too much work and of little value to customers. However, there has been increasing awareness of the importance of architectural practices in agile development. A lack of architecture leads to the inability to reuse code components, structural problems and maintenance problems of the software system. There has also been report of quality issues in the software built using agile methods. This is reportedly due to the constraint of time (sprints), frequent releases and the obligation to have working software in a short time, which can lead to the accumulation of quality issues [26] [24]. A summary of the shortcomings and challenges with agile processes are shown below in Table 1.

TABLE I
Shortcomings and Challenges of Agile Processes

Challenges with Agile
Difficult to estimate effort based on the story cards and preliminary customer feedback. [27] [28] [29]
Little or no documentation [21] [20] [22] [23] [30] [31] [26]
Story cards are usually incomplete, vague, or lack detail. Story cards inadequate for testing specifications, do not account for re-factoring and are liable to being lost. [17] [18]
Non-functional requirements often ignored. No systematic way to address requirements dependencies. Difficult to prioritize user stories. Insufficient requirement verification. [18] [17] [32] [33]
Simple design equated to least time design, too abstract or not practiced at all. [18]
Lack of or insufficient architectural design [18] [24] [25] [34]
Does not support the building of reusable artefacts [30] [35] [24] [31]
Low level of test coverage [30] [33] [26]
Quality issues due to time, budget and functionality constraints. [24] [26]

Yet despite their shortcomings agile processes can provide a competitive advantage for small companies since their small releases provide focus and faster delivery [5]. For economic sustainability small companies need mechanisms for cost saving and for prudent use of resources.

B. Process Tailoring

Although prescriptive descriptions of software development methodologies exist, in practice software development methodologies are often times adjusted or tailored according to the needs for the project being implemented [36] [37] [38]. According to [36] it is unusual for a software method to be used completely in its textbook form, instead adjustments are made, tailoring to suit the project being executed. Even quality models like CMMI and ISO/IEC 12007 consider tailoring of the process as a requirement for software development companies [38]. Yet there is little evidence to show tailoring done to specifically address the shortcomings of the agile process.

Several examples are provided in the literature of tailored agile processes. In [39] a hybrid of XP, Scrum and RUP is proposed. The aim was to integrate the best components (strengths) of these models while reducing their limitations thus providing a model that will produce quality software in a timely and cost effective manner [39]. Each of XP, Scrum and RUP model, would provide some important characteristic to the new hybrid model. XP would provide the necessary software engineering practices, Scrum the managerial structure and RUP the plan driven and documentation structure [39]. However, the proposal lacked details about which specific elements of the models would be included and which would be left out of the hybrid. Furthermore, the description of the hybrid is not sufficiently detailed for others to follow and use the proposed hybrid model; even though according to the authors the hybrid has not been tested and will need to be tested, by other developers, in a live environment [39].

Cao [40] investigated the issue of how agile methodologies are adapted for various contexts. The study used adaptive structural theory (AST) as the theoretical basis on which to examine the issue. The AST theory links social structure to information technologies structures, processes and organizations [40]. The research involved multiple case studies of four projects executed in different companies under varying conditions. Cao [40] found that for the small project agile methodology could be used without adjustments, however, for the larger more complex projects the agile practices had to be adjusted in order to efficaciously execute the projects. In [40], like [41], some waterfall practices were incorporated for the purpose of better managing the larger projects. The primary adjustments were abstracted architectural designs, design and development for reuse, formal agreements for design and requirements, post hoc and relevant documentation, pair programming for some aspects or no pair programming at all, rotation of team members and roles, shared expertise, agreement on quality and formal costs and estimates [40].

In [42] the researcher used a case study approach to examine a hybrid of RUP and Scrum. Castilla [42] like [39] began with the posit that by combining the methodologies their strengths can be maximized and their weaknesses reduced, thus providing a more optimize process to produce a

high quality product. In this hybrid the RUP provide the overarching framework, structure and guide for the development process, while SCRUM provided the day-to-day management and organization for project [42]. RUP was used for the first two stages, (Inception and Elaboration), of the process and Scrum was used for the last two stages, (Constuction and Transition), as well as a part of the Elaboration stage of the process. The hybrid provided several benefit, for example using RUP's approach during the first part of the project provided a common and accurate understanding of the organization and project requirements in addition customer's response and input to project documents and models could be accommodated due to RUP's incremental and iterative approach [42]. In addition using Scrum meant that sprint goals could be met, yielding frequent product releases, with timely customer feedback, avoiding delays due to changing requirements late in the project and hence reducing risk and project duration [42].

Mushtaq et al [43] presented a hybrid agile process, which combined XP engineering features with Scrum managerial features. The rationale behind the hybrid is that Scrum's effective project management framework can be enhanced with XP's product development capabilities, thus providing a process model that can be used to provide high quality software for varying size projects [43]. The hybrid was validated using a controlled case study.

Our project involves a different approach that of addressing shortcomings that are common across several agile processes by tailoring the specific element or activity in the process. The two primary shortcomings identified across the agile processes are (1) the lack of software architecture and (2) the lack of detailed requirements. The general structure of the processes is maintained with the relevant adjustment added at the appropriate stage.

III. Methodology

The altered and unaltered processes were evaluated using a controlled case study in a similar manner as in [44]. Controlled case study is an approach that combines different aspects of case study, experiment and action research [45] [46]. It involves carrying out a project aimed to deliver a software product to client(s) while constantly collecting metrics [45]. In this case some of the experimental features in our project were the randomly selected team members and the random assignment of process to each team. The action research features involved the weekly meetings to assess the project status, to make adjustments were necessary and to guide teams so that they remained true to the process. The project instance provided the case study setting for the study. Metrics related to efficiency, quality and cost effectiveness of the process were collected for each process throughout the execution of the project.

The shortcomings addressed were (1) the lack of an architectural design in some agile processes; and (2) the inadequately detailed user stories i.e. poor user requirements. Three agile processes, namely Feature Driven Development (FDD), SCRUM and Extreme Programming (XP) were tailored. As a measure of control and for comparison, data was gathered on the tailored processes and the untailored processes which were carried out by different sets of student teams. Detailed requirements and architectural design were added to the processes that lacked them. Thus detailed requirement steps were added to Feature Driven Development, SCRUM and Extreme Programming, while architectural design step was added to Extreme Programming.

The controlled case study was conducted in an academic environment, using students enrolled in a software development course. The students were randomly assigned to development teams. Each team was then randomly assigned one of the unaltered agile method or a tailored agile method. The agile methods used were (1) Feature Driven Development (FDD), (2) Feature Driven Development tailored with detailed requirements (FDD_DReq), (3) Scrum (SCRUM), (4) Scrum tailored with detailed requirements (SCRUM_DReq), (5) Extreme programming (XP), Extreme programming tailored with detailed requirements (XP_DReq), and Extreme programming tailored with architectural design (XP_AD). Teams were required to develop the same software product using the method they were assigned. All teams used the same programming language for development. The students were trained with respect to their assigned process and given written guidelines. In addition weekly meetings were held to coach team in terms of their assigned process. All groups were given the same product requirements. Product development lasted for five weeks, using one week sprints. The teams were asked to develop the software product using the development method they were randomly assigned.

Weekly meetings were held with each team. The primary purpose of these meetings was to determine progress made, the plan for the coming week, and to evaluate and correct any deviation from the prescribe process. Data was gathered on the process on a weekly basis and the software artefacts generated. Several metrics were used to evaluate the processes. Emphasis was placed on the defect metrics of the code since these were objective and factual measures and the alternative time measures were self reported and therefore subject to bias and memory. Efficiency was measured using rate of change from version to version and defects per effort (effort measured in lines of code and function point). Product quality was measured in terms of number of defects and defects per line of code. Cost effectiveness was measured in terms of lines of code per person month and lines of code per function point. The metrics were then analysed, and used to identify the most efficient process, the process that produced the highest quality product and the most cost effective process.

IV. Findings and Analysis

Of the six processes used, the product developed using FDD had the least number of defects with 9, followed by the product of the FDD_DReq process with 14.

TABLE 2
MEASURES OF EFFICIENCY

Process	Defects	LOC	Warnings	Man month	Defect / LOC per Function Point	Defect / LOC per man month
FDD	9	2007	24	0.468	2.76809E-05	0.009643667
FDD_DReq	14	1476	16	0.468	5.855E-05	0.020398053
SCRUM	22	3223	101	0.468	4.21354E-05	0.014679438
SCRUM_DReq	21	1862	5	0.468	6.96185E-05	0.024254184
XP	18	1634	28	0.468	6.79995E-05	0.023690133
XP_DReq	30	1637	16	0.468	1.13125E-4	0.039411197
XP_AD	15	1739	17	0.468	5.3245E-05	0.01854978

The product output from the SCRUM process had the most warnings with 101, but not the most defects. On the other hand the product generated from SCRUM_DReq had the least number of warnings with 5, but a relatively high number of 21 defects when compared to the other processes. XP_DReq had a comparable number of warnings with 16 compared to the others, but a relatively high number of defects with 30 again when compared with the others. In fact XP_DReq was the process with the highest number of defects. The SCRUM generated product had the most lines of code. The product generated using FDD had the least defect per line of code per function point with 2.77×10^{-5} as well as the least defect per line of code per man month with 9.6×10^{-3}. FDD was the most efficient process followed by SCRUM. Details are shown in Table 2 above.

Quality was measured in terms of defects per function point and defects per thousand lines of code. The code and product generated using the FDD method had the lowest number of defects with 9, lowest defects per function points of 0.06 and the lowest defects per thousand lines of code of 4.48. This was followed by the code generated by SCRUM.

TABLE 3
MEASURES OF QUALITY

Process	Defects	LOC	Defect per function point	Defect per KLOC final version
FDD	9	2007	0.06	4.48
FDD_DReq	14	1476	0.09	9.49
SCRUM	22	3223	0.14	6.83
SCRUM_DReq	21	1862	0.13	11.28
XP	18	1634	0.11	11.02
XP_DReq	30	1637	0.19	18.33
XP_AD	15	1739	0.09	8.63

On the other hand the output from XP_DReq had the highest number of defects, defects per function point of 0.19 and defects per thousand lines of code of 18.33. The products generated from FDD_DReq and XP_AD had similar defects per function point of 0.09 yet differed in terms of defects per thousand lines of code with 9.49 and 8.63 respectively. These results are shown in Table 3 above.

In terms of cost effectiveness the SCRUM process had the highest lines of code per person month with 6886 and per function point with 19.90 indicating that the SCRUM process was the most cost effective.

TABLE 4
MEASURES OF COST EFFECTIVENESS

Process	LOC	Man month	LOC per person month	LOC per function point
FDD	2007	0.468	4288.46	12.39
FDD_DReq	1476	0.468	3153.85	9.11
SCRUM	3223	0.468	6886.75	19.90
SCRUM_DReq	1862	0.468	3978.63	11.49
XP	1634	0.468	3491.45	10.09
XP_DReq	1637	0.468	3497.86	10.10
XP_AD	1739	0.468	3715.81	10.73

This was followed by FDD with 4288 lines of code per person month and then SCRUM_DReq with 3978. It appears that SCRUM processes (altered and unaltered) are fairly cost effective. FDD_DReq was the least cost effective with 9.11 for lines of code per function point. The cost effectiveness of the three extreme programming processes were fairly similar, i.e. 10.09, 10.10 and 10.73 in terms of lines of code per function point this is interesting since none of the other processes had a similar pattern. This may be an interesting angle to pursue further investigation of extreme programming with respect to economic sustainability in small software development companies. The detailed for the cost effectiveness metrics results are shown in Table 4 above.

Of all the processes only the altered XP_AD process produced better results for efficiency. This is in keeping with the literature [18] [24] [25] and [34] that indicate that an architectural design provides general direction for the product team by giving an overview of the general project direction. This result may be in support of the upfront architectural design in extreme processing process. The altered processes FDD_DReq and XP_AD had similar quality measures in terms of defects per function point. This could be because detailed requirements and up front architectural design can help to clarify software features and functionalities. Both the altered and unaltered SCRUM processes had similar measures which could mean that quality in terms of function points for SCRUM can be fairly consistent. So there is some indication of improved quality with more detailed requirements. Surprisingly, except for extreme programming the unaltered processes are more cost effective in terms of lines of code per function point. Perhaps this is an indication that making changes to the process may require the use of more resources.

V. CONCLUSION

Altering an agile process can produce some useful benefits. The FDD process seemed to have outperformed all the others processes. In addition the altered process

FDD_DReq performed fairly well. This is in keeping with the nature of feature driven development. Architectural design and detailed requirement can help to produce greater quality software. The altered versions of extreme programming were more cost effective, and this may be an area for further investigation especially with respect to improving economic sustainability in software development. The project could be repeated to verify these results.

REFERENCES

[1] M. Höjer and J. Wangel, "Smart Sustainable Cities: Definition and Challenges". In Hilty, L.M., Aebischer, B. (eds) ICT Innovation for Sustainability. Advances in Intelligent Systems and Computing: Springer International Publishing, 2014. pp. 333-349.

[2] F. Russo and A. Comi, "City characteristics and urban goods movements: A way to environmental transportation system in a sustainable city". Procedia - Social and Behavioral Sciences, vol 39, 2012. pp. 61–73.

[3] N. Khansari, A. Mastashari and M. Mansouri, "Impacting Sustainable Behaviour and Planning in Smart City", International Journal of Sustainable Land Use and Urban Planning. Vol. 1 No. 2, 2013. pp. 46-61.

[4] T. Klein, G. Reinhart, "Towards agile engineering of mechatronic systems in machinery and plant construction". Procedia Cirp 2016, vol 52. pp. 68–73.

[5] Z. Galvina, D. Smite, "Software Development Processes in Globally Distributed Environment", Scientific Papers, University of Latvia, 2011. Vol. 770, Computer Science and Information Technologies

[6] L. Williams and A. Cockburn, "Agile Software Development: It's about Feedback and Change", Computer, IEEE Computer Society, June 2003, pp. 39-43

[7] G. Kapitsaki and M. Christou, "Learning from the Current Status of Agile Adoption" in J. Filipe and L. Maciaszek (Eds.): ENASE 2014, CCIS 551, 2015. pp. 18–32.

[8] M. Rychl´y and P. Tich´a, "A Tool for Supporting Feature-Driven Development" in *Balancing Agility and Formalism in Software Engineering*, eds B. Meyer, J.R. Nawrocki, and B. Walter, Springer, Germany, 2008. pp. 196–207

[9] C. Ferreira and J. Cohen, "Agile Systems Development and Stakeholder Satisfaction: A South African Empirical Study", SAICSIT 2008, October 2008,

[10] K. Rao, G. Naidu, and P. Chakka, "A study of the Agile Software Development Methods", Applicability and Implications in Industry, in International Journal of Software Engineering and Its Applications, vol. 5 no. 2, April 2011

[11] D. Turk, R. France and B. Rumpe, "Assumptions Underlying Software Development Processes", in Journal of Database Management, vol 16, no. 4, 2005, pp 62-87

[12] Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J., Agile software development methods. Review and analysis. Espoo 2002. VTT Publications 478

[13] V. Moreno and J. Pinheiro, "Influence of Technical and Management Capacities on the Performance of Brazilian Software Development Firms" in Proceedings Annual Workshop of the AIS Special Interest Group for ICT in Global Development. 2010

[14] F. Kamei, G. Pinto, B. Cartaxo and A. Vasconceios, "On the Benefits/Limitations of Agile Software Development: An Interview Study with Brazilian Companies", in Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE), June 2017

[15] K. Petersen and C. Wohlin, "A comparison of issues and advantages in agile and incremental development between state of art and an industrial case", Journal of Systems and Software, 2009

[16] R. Shankarmani, R. Pawar, S. Mantha and V. Babu, "Agile Methodology Adoption: Benefits and Constraints". International Journal of Computer Applications. vol 58. 2012. pp. 31-37

[17] A. Patel, A. Seyfi, M. Taghavi, C. Wills, L. Na, R. Latih, S. Misra, "A Comparative Study of Agile, Component-Based, Aspect-Oriented and Mashup Software Development Methods", Technical Gazette, 19(1), 2012. pp. 175-189.

[18] B. Kongyai and E. Edi, "Adaptation of Agile Practices : A Systematic Review and Survey". 2011

[19] M. Qureshi, J., Ikram, "Proposal of Enhanced Extreme Programming Model", I.J. Information Engineering and Electronic Business, vol 1 2015. pp. 37-42

[20] N. Uikey, U, Suman, A. Ramani, "A Documanted Approach in Agile Software Development", International Journal of Software Engineering (IJSE), vol. 2 issue 2, 2011. pp. 13-22

[21] A. Moniruzzaman, and S. Hossain, "Comparative Study on Agile Software Development Methodologies", Global Journal of Computer Science and Technology, vol. 13 issue 8, 2013. pp. 5-18.

[22] R. Hoda, J. Noble and S. Marshall, "Documentation strategies on agile software development projects", International Journal on Agile and Extreme Software Development, Vol 1, No. 1, 2012. pp. 23-37, 2012

[23] J. Nuottila, K. Aaltonen, and J. Kujala, "Challenges of adoptin agile methods I a public organization", International Journal of Information Systems and Project Management, Vol. 4, No. 3, 2016, pp 65-85

[24] C. Álvarez, "Overcoming the Limitations of Agile Software Development and Software Architecture". Master's Thesis, Blekinge Institute of Technology, September 2013.

[25] M. Babar, "An Exploratory Study of Architectural Practices and Challenges in Using Agile Software Development Approaches", in Joint Working IEEE/IFIP Conf. on Software Architecture & European Conf. on Software Architecture, 2009. pp. 81-90.

[26] A. Majeed, "Issues and Challenges in Scrum Implementation", International Journal of Scientific & Engineering Research, Vol 3, No. 8, Aug 2012. pp. 1-4

[27] A. Raslan, N. Darwish, (2018), "An Enhanced Framework for Effort Estimation of Agile Projects", International Journal of Intelligent Engineering and Systems, Vol. 11, No. 3, 2018. pp 205-214

[28] M. Abouelela, L. Benedicenti, L., (2010), Bayesian Network Based XP Process Modelling, International Journal of Software Engineering & Applications, Vol. 1, No. 3, 2010. pp. 1-15

[29] K. Jammalamadaka, R. Krishna, "Agile Software Development and Challenges", International Journal of Research in Engineering and Technology, Vol 02, Issue 08, Aug. 2013. pp. 125-129

[30] T. Sekgweleo, "Understanding Agile System Development Methodologies", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 5, Issue 7, July 2015. pp. 18-24

[31] Y. Leau, W. Loo, W. Than, S. Tan, "Software Development Life Cycle Agile vs. Traditional Approaches", International Conference on Information and Network Technology, IPCSIT, vol. 37, 2012. pp. 162-167

[32] A. Martakis, M. Daneva, "Handling requirements dependencies in agile projects: A focus group with agile software development practitioners", IEEE 7th International Conference on Research Challenges in Information Science (RCIS), 2013. pp. 1-11.

[33] R. Anand. M. Dinakaran, "Issues in Scrum Agile Development Principles and Practices in Software Development", Indian Journal of Science and Technology, Vol. 8 (35), 2015. pp. 1-5

[34] C. Prause, Z. Durdik, "Architectural Design and Documentation: Waste in Agile Development?", in Proceedings of 2012 International Conference on Software and System Process (ICSSP), Zurich, Switzerland, 2012. pp. 130-134

[35] A. Mahanti, "Challenges in Enterprise Adoption of Agile Methods – A Survey", Journal of Computing and Information Technology (CIT), Vol. 14 No. 3, 2006. pp. 197-206

[36] K. Conboy and B. Fitzgerald, "Method and Developer Characteristics for Effective Agile Method Tailoring: A Study of XP Expert Opinion", ACM Transactions on Software Engineering and Methodology, Vol. 20 No. 1, Article 2, 2010. pp. 2:1-2:30

[37] A. Campanelli, "A Tailoring Criteria Model for Agile Practices Adoption", Masters Thesis, Universidada Fumec, 2015

[38] R. Santos, T. Oliveira, F. Abreu, "Mining Software Development Variations", *SAC'15*, , Salamanca, Spain. April 13-17, 2015.

[39] G. Ahmad, T. Soomro and M. Brohi, "XSR: Novel Hybrid Software Development Model (Integrating XP, Scrum & RUP)", International Journal of Soft Computing and Engineering (IJSCE), Vol. 2 Issue 3, 2014. pp. 126-130

[40] L. Cao, K. Mohan, P. Xu, B. Ramesh, "A Framework for Adopting Agile Development Methodologies", in European Journal of Information Systems, vol 18, 2009. pp. 332-343

[41]E. Burman, “Agile in Action: Hybrid Methodologies in Practise”. Master Thesis, Umeå, 2015
[42]D. Castilla, “A Hybrid Approach Using RUP and Scrum as a Software Development Strategy”, Masters Thesis, University of North Florida, 2014
[43]Z. Mushtaq, M. Qureshi. "Novel Hybrid Model: Integrating Scrum and XP", IJITCS, vol.4, no.6, 2012. pp. 39-44
[44]G. Rasool, S. Aftab, S. Hussain and D. Streitferdt, “eXRUP: A Hybrid Software Development Model for Small to Medium Scale Projects”, Journal of Software Engineering and Applications, vol 6, 2013. pp. 446-457
[45]O. Salo and P., Abrahamsson, “Empirical Evaluation of Agile Software Development: the Controlled Case Study Approach”, in the Proceedings of the 5th International Conference on Product Focused Software Process Improvement, Keihanna-Plaza, Kansai Science City in Kyoto-Nara, Japan
[46]M. Siniaalto, P. Abrahamsson, P. “A comparative case study on the impact of test-driven development on program design and test coverage”. In Proceedings of First International Symposium on Empirical Software Engineering and Measurement (ESEM), 2007. pp. 275-284.
.