A Critical Review of Agile Development Methods

[1]Amit Sharma, [2]Dr R.K Bawa
[1]Faculty of Science & Technology, ICFAI University, Baddi, India
[2]Dept. of Computer Science, Punjabi University, Punjab, India

Abstract
Looking at the software engineering principles from a historical perspective, we can see how the software processing methodologies evolved since past 50 years, but probably the most discernible exchange to software business in recent years has been the introduction of evince "Agile". Agile software development has been used by industry to create a more flexible and lean software development process, i.e making it possible to develop software at a faster rate and with more agility during development. Popular methods in the agile ecosystem have a broad range of characteristics. Methods like SCRUM, Extreme programming (XP), Feature driven Development (FDD), Adaptive software development (ASD) etc are increasingly being used to develop software using an adaptation approach rather than a predictive one. But, there is a scarcity of the resources which describe on how these resources can be integrated with the agile methodologies. This paper basically reviews different agile methods, how they are divergent from the conventional process methods, the pros and cons of applying agile processes to the research projects, and what are the difficulties faced during enforcement of agile methodology in the project.

Keywords
Agile Software Development, Extreme Programming, SCRUM, Feature driven development.

I. Introduction

Plan-driven methods are those that begin with the solicitation and documentation of a set of requirements that is as complete as possible. Based on these requirements, one can then formulate a plan of development. Usually, the more complete the requirements, the better the plan. Some examples of plan-driven methods are various waterfall approaches and others such as the Personal Software Process (PSP) and the Rational Unified Process (RUP). An underlying assumption in plan-driven processes is that the requirements are relatively static. On the other hand, iterative methods, such as spiral model based approaches, evolutionary processes described in and recently agile approaches count on change and recognize that the only constant is change. The question is only of the degree and the impact of the change. Beginning in the mid-1990's, practitioners began finding the rate of change in software requirements increasing well beyond the capabilities of classical development methodologies. The software industry, software technology, and customers expectations were moving very quickly and the customers were becoming increasingly less able to fully state their needs up front. As a result, agile methodologies and practices emerged as an explicit attempt to more formally embrace higher rates of requirements change.

A. Background

Agile methods are a subset of iterative and evolutionary methods and are based on iterative enhancement and opportunistic development processes. In all iterative products, each iteration is a self-contained, mini-project with activities that span requirements analysis, design, implementation, and test. Each iteration leads to an iteration release (which may be only an internal release) that integrates all software across the team and is a growing and evolving subset of the final system. The purpose of having short iterations is so that feedback from iterations N and earlier, and any other new information, can lead to refinement and requirements adaptation for iteration N + 1. The customer adaptively specifies his or her requirements for the next release based on observation of the evolving product, rather than speculation at the start of the project. There is quantitative evidence that frequent deadlines reduce the variance of a software process and, thus, may increase its predictability and efficiency.

B. Agile Manifesto

In February 2001, several software engineering consultants joined forces and began to classify a number of similar change-sensitive methodologies as agile (a term with a decade of use in flexible manufacturing practices [34] which began to be used for software development in the late 1990's). The term promoted the professed ability for rapid and flexible response to change of the methodologies. The consultants formed the Agile Alliance and wrote The Manifesto for Agile Software Development and the Principles behind the Agile Manifesto [1]. The methodologies originally embraced by the Agile Alliance were Adaptive Software Development (ASD), Crystal, Dynamic Systems Development Method (DSDM), Extreme Programming (XP), Feature Driven Development (FDD) and Scrum.
The manifesto reads as follows (Agile Alliance, 2001): "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value [15]

- Individuals and interactions over Processes and tool
- Working software over Comprehensive documentation
- Customer collaboration over Contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more". [1]

Table 1: Traditional and Agile Perspectives on Software Development

	Traditional view	Agile perspective
Design process	Deliberate and formal, linear sequence of steps, separate formulation and implementation, rule-driven	Emergent, iterative and exploratory, knowing and action inseparable, beyond formal rules
Goal	Optimization	Adaptation, flexibility, responsiveness
Problem-solving process	Selection of the best means to accom- plish a given end through well-planned, formalized activities	Learning through experimentation and introspection, constantly reframing the problem and its solution

View of the environment	Stable, predictable	Turbulent, difficult to predict
Type of learning	Single-loop/adaptive	Double-loop/ generative
Key characteristics	Control and direction Avoids conflict Formalizes innovation Manager is controller Design precedes implementation	Collaboration and communication; integrates different worldviews Embraces conflict and dialectics Encourages exploration and creativity; opportunistic Manager is facilitator Design and implementation are inseparable and evolve iteratively
Rationality	Technical/functional	Substantial
Theoretical and/or philosophical roots	Logical positivism, scientific method	Action learning, John Dewey's pragmatism, phenomenology

The previous four values have been further defined by twelve principles: [1]

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes tackle change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

II. An Overview of Agile Methods

In this section the existing agile methods are identified and their objectives briefly introduced. Fig. 1 shows the manifold agile software development methods and their interrelationships together with their evolutionary paths. Fig. 1, purposefully, extends the considerations beyond the scope of this paper. This means that some, more philosophical meta-level expositions, which have in their turn impinged upon the preceding agile methods either directly or indirectly are included in the diagram. Figure 1 also depicts (i.e., using a dashed line) which methods (or method developers) contributed to the publication of the agile manifesto. (http://www.agilemanifesto.org).

For the purposes of this paper, agile software development in general is characterized by the following attributes: incremental, cooperative, straightforward and adaptive [11]. Incremental refers to small software releases, with rapid development cycles. Cooperative refers to a close customer and developer interaction. Straightforward implies that the method itself is easy to learn and to modify and that it is sufficiently documented.

A. Extreme Programming (XP)

Extreme Programming (XP) [6,20,21] originators aimed at developing a methodology suitable for "object-oriented projects using teams of a dozen or fewer programmers in one location."

The methodology is based upon five underlying values: communication, simplicity, feedback, courage, and respect.

1. Communication

XP has a culture of oral communication and its practices are designed to encourage interaction. The communication value is based on the observation that most project difficulties occur because someone should have spoken with someone else to clarify a question, collaborate, or obtain help. "Problems with projects can invariably be traced back to somebody not talking to somebody else about something important."

2. Simplicity

Design the simplest product that meets the customer's needs. An important aspect of the value is to only design and code what is in the current requirements rather than to anticipate and plan for unstated requirements.

3. Feedback

The development team obtains feedback from the customers at the end of each iteration and external release. This feedback drives the next iteration. Additionally, there are very short design and implementation feedback loops built into the methodology via pair programming and test-driven development.

4. Courage

The other three values allow the team to have courage in its actions and decision making. For example, the development team might have the courage to resist pressure to make unrealistic commitments.

5. Respect

Team members need to care about each other and about the project.

B. Crystal Clear

Crystal Clear [15-17] is targeted at a D6 project and could be applied to a C6 or a E6 project and possibly to a D10 project.

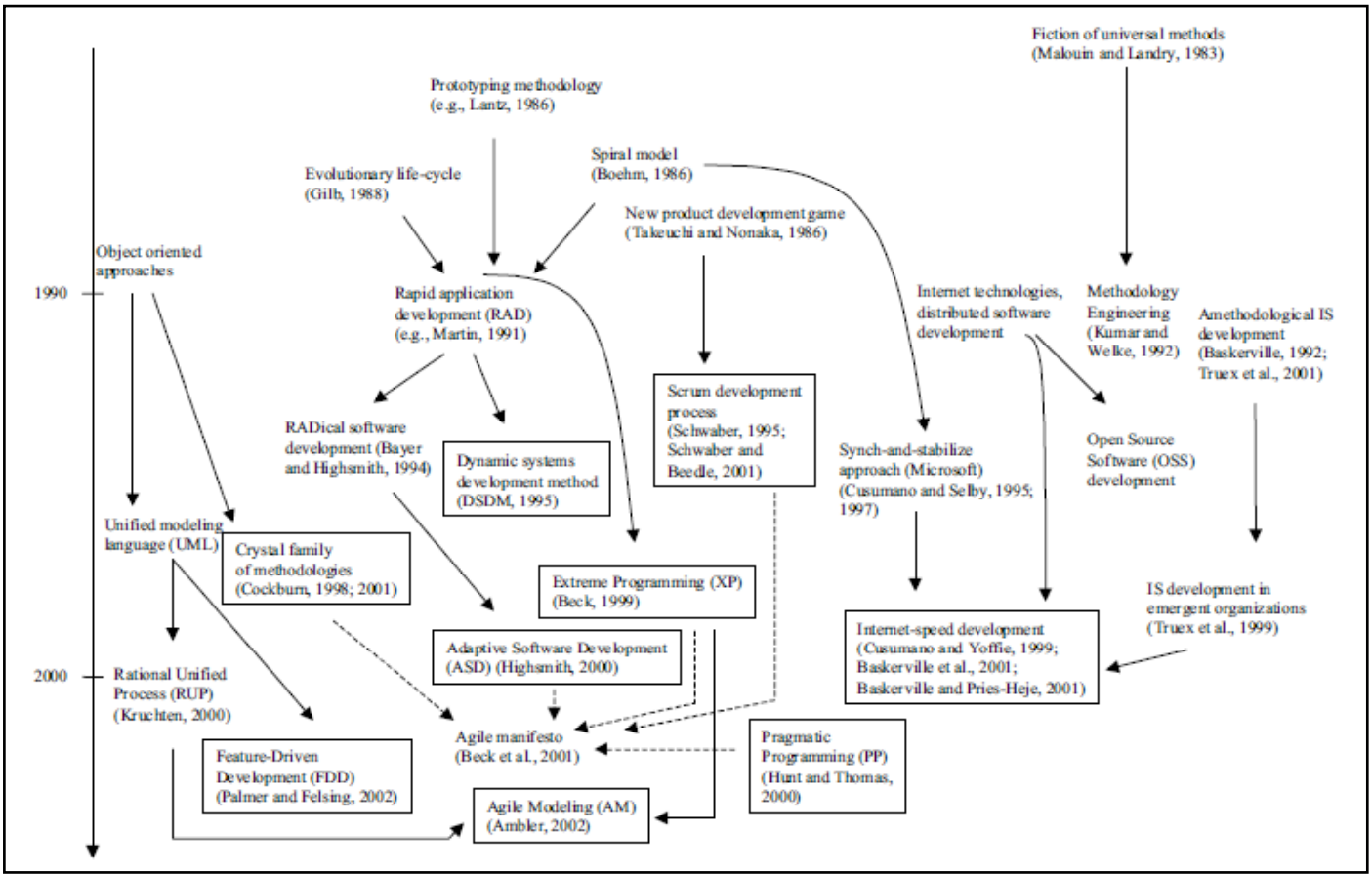

Fig. 1: Evolutionary Map of Agile Methods

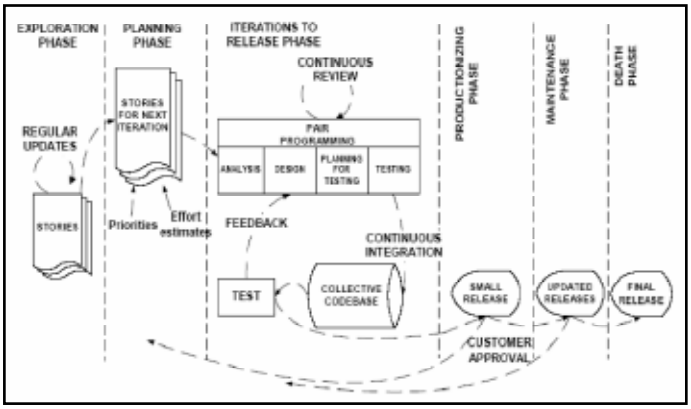

Fig. 2: Lifecycle of the XP Process

Crystal Clear is an optimization of Crystal that can be applied when the team consists of three to eight people sitting in the same room or adjoining offices. The property of close communication is strengthened to "osmotic" communication meaning that people overhear each other discussing project priorities, status, requirements, and design on a daily basis. Crystal Clear's model elements are as follows:

- **Documents and Artifacts:** Release plan, schedule of reviews, informal/low-ceremony use cases, design sketches, running code, common object model, test cases, and user manual
- **Roles:** Project sponsor/customer, senior designer-programmer, designer programmer and user (part time at least).
- **Process:** Incremental delivery, releases less than two to three months, some automated testing, direct user involvement, two user reviews per release, and methodology-tuning retrospectives. Progress is tracked by software delivered or major decisions reached, not by documents completed.

C. Scrum

In the Scrum process [32-33] puts a project management "wrapper" around software development methodology. The methodology is flexible on how much/how little ceremony but the Scrum philosophy would guide a team towards as little ceremony as possible. Usually a Scrum teams works co-located. However, there have been Scrum teams that work geographically distributed whereby team members participate in daily meeting via speakerphone. Scrum teams are self-directed and self-organizing teams. The team commits to a defined goal for an iteration and is given the authority, autonomy, and responsibility to decide how best to meet it.

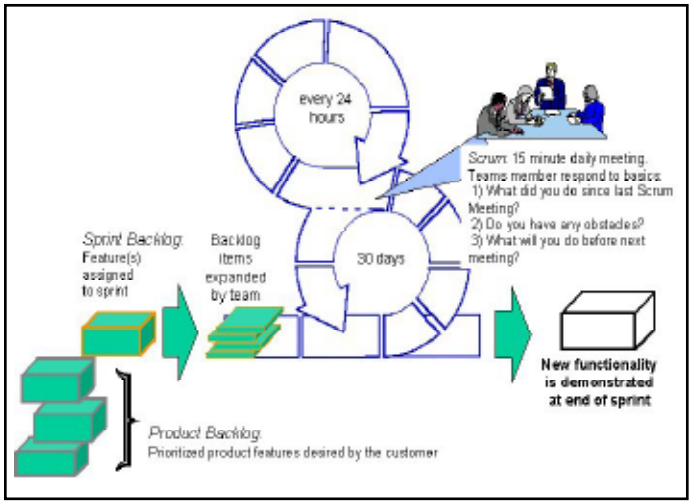

Fig. 4: The Scrum Process

D. Feature-Driven Development

Feature Driven Development (FDD) [22-23] authors Peter Coad and Jeff de Luca characterizes the methodology as having

"just enough process to ensure scalability and repeatability and encourage creativity and innovation all along the way." Throughout, FDD emphasizes the importance of having good people and strong domain experts. FDD is build around eight best practices: domain object modeling; developing by feature; individual class ownership; feature teams; inspections; regular builds; configuration management; reporting/visibility of results. UML models [24] are used extensively in FDD.

1. Documents and Artifacts

- **Feature lists,** consisting of a set of features whereby features are small, useful in the eyes of the client, results; a client-valued function that can be implemented in two weeks or less. If a feature would take more than two weeks to implement, it must be further decomposed.
- **Design packages** consist of sequence diagrams and class diagrams and method design information
- **Track by Feature**, a chart which enumerates the features that are to be built and the dates when each milestone has been completed.
- **"Burn Up" Chart,** a chart that has dates (time) on the x axis. On the y axis is an increasing number of features that have been completed. As features are completed this chart indicates a positive slope over time.

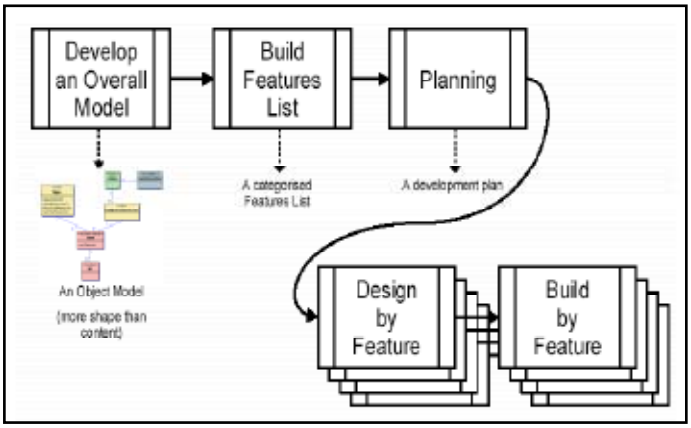

Fig. 5: Feature Driven Development Process

E. Lean Development

The most strategic-oriented ASDE is also the least known: Bob Charette's Lean Development, which is derived from the principles of lean production, the restructuring of the Japanese automobile industry that occurred in the 1980's. In LD, Bob extends traditional methodology's view of change as a risk of loss to be controlled with restrictive management practices to a view of change producing "opportunities" to be pursued during "risk entrepreneurship". LD has been used successfully on a number of large telecommunications projects in Europe.

F. Adaptive Software Development

Adaptive Software Development, [12] contribution to the Agile movement, provides a philosophical background for Agile methods, showing how software development organisations can respond to the turbulence of the current business climate by harnessing rather than avoiding change. ASD contains both practices – iterative development, feature-based planning, customer focus group reviews – and an "Agile" management philosophy called Leadership- Collaboration management. One ancestor of ASD is "RADical Software Development" [13]. ASD claims to provide a framework with enough guidance to prevent projects from falling into chaos, but not too much, which could suppress emergence and creativity.

G. Dynamic Systems Development Method

Dynamic systems development method (DSDM) [18-19] is a method developed by a dedicated consortium in the UK. The first release of the method was in 1994. The fundamental idea behind DSDM is that instead of fixing the amount of functionality in a product, and then adjusting time and resources to reach that functionality, it is preferred to fix time and resources, and then adjust the amount of functionality accordingly. The origins of DSDM are in rapid application development. DSDM can be seen as the first truly agile software development method.

III. Summary

This section summarizes the agile methods discussed in the previous section by comparing these methods as shown in Table II and at the end general features of these agile methods are summarized in Table 3.

Table 2: Comparison of Agile Methods

Concept	XP	SCRUM	DSDM	CRYSTAL	FDD
Team size	3-16	5-9	2-6	4-8	6-15
Number of teams	1	1-4	1-6	1-10	1-3
Volatility	high	high	low	high	low
Team distribution	no	no	yes	yes	yes

Table 3: General Features of Agile Methods

Method Name	Key Points	Special features	Identified weakness
ASD	Adaptive culture, collaboration, mission-driven component based iterative development	Organizations are seen as adaptive systems. Creating an emergent order out of a web of interconnected individuals	ASD is more about concepts and culture than the software practice
DSDM	Application of controls to RAD, use of timeboxing and empowered DSDM teams.	First truly agile software development method, use of prototyping, several user roles : "ambassador", "visionary" and "advisor"	While the method is available, only consortium members have access to white papers dealing with the actual use of the method
XP	Customer driven development, small teams, daily builds	Refactoring - the ongoing redesign of the system to improve its performance and responsiveness too change	While individual practices are suitable for many situations, overall view & management practices are given less attention
SCRUM	Independent, small, self-organizing development teams, 30-day release cycles.	Enforce a paradigm shift from the "defined and repeatable" to the "new product development view of Scrum"	While Scrum details in specific how to manage the 30-day release cycle, the integration and acceptance tests are not detailed
FDD	Five-step process, object-oriented component (i.e. feature) based development.	Method simplicity, design and implement the system by features, object modeling	FDD focuses only on design and implementation. Needs other supporting approaches.

IV. Conclusion

In this paper, we presented our analysis of different agile software methods. We also describe what are the problems faced during implementation of agile software development. The objective is to help software engineers to understand the key characteristics of these methods and therefore select the most suitable method with respect to the type of software projects they develop. As a future work, there is a need to review other agile processes not covered in this paper such as the Distributed Agile Development (DASD) etc.

References

[1] Beck K., et al. Manifesto for Agile Software Development, February 2001.

[2] J. Highsmith, "The great methodologies debate: Part2," Cutter IT Journal, vol. 15, 2002.

[3] J. Highsmith, "The great methodologies debate: Part1," Cutter IT Journal, vol. 14, 2001.

[4] E. Yourdon,"Light methodologies," Cutter IT Journal, vol. 13, 2000.

[5] R. McCauley, "Agile Development Methods Poised to Upset Status Quo," SIGCSE Bulletin, Vol. 33, pp. 14 - 15, 2001.

[6] K. Beck,"Embracing Change With Extreme Programming," IEEE Computer, Vol. 32, pp. 70-77,1999.

[7] J. Highsmith and A. Cockburn, "Agile Software Development: The Business of Innovation," Computer, Vol. 34, pp. 120-122, 2001.

[8] B. Boehm, "Get Ready For The Agile Methods, With Care," Computer, Vol. 35, pp. 64-69, 2002.

[9] M. Fowler and J. Highsmith, "Agile methodologists agree on something," Software Development, Vol. 9, pp. 28-32, 2001.

[10] M. M. Müller, W. F. Tichy,"Case Study: Extreme Programming in a University Environment," Presented at 23rd International Conference on Software Engineering, Toronto, 2001.

[11] P. Abrahamsson, O. Salo, J. Ronkainen, J.Warsta, Agile software development methods: Review and Analysis. Espoo, Finland: Technical Research Centre of Finland, VTT Publications 478, Research Centre of Finland, VTT Publications 478, [Online] Available: http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf,2002.

[12] J. A. Highsmith, Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. New York, NY: Dorset House Publishing, 2000.

[13] S. Bayer, J. Highsmith, "RADical software development", American Programmer, Vol. 7, pp. 35-42, 1994.

[14] S. Ambler, Agile Modeling:Effective Practices for Extreme Programming and the Unified Process. New York: John Wiley & Sons, Inc. New York, 2002.

[15] A. Cockburn, Surviving Object-Oriented Wesley Longman,1998.

[16] A. Cockburn, Writing Effective Use Cases, The Crystal Collection for Software Professionals: Addison-Wesley Professional, 2000.

[17] A. Cockburn, Agile Software Development. Boston: Addison-Wesley, 2002.

[18] DSDMConsortium, Dynamic Systems Development Method, version 3. Ashford, Eng.: DSDM Consortium, 1997.

[19] J. Stapleton, Dynamic systems development method - The method in practice: Addison Wesley, 1997.

[20] K. Beck, Extreme programming explained. Reading, Mass.: Addison-Wesley, 1999.

[21] K. Beck,"Extreme Programming Explained: Embrace Change, 2000.

[22] S. R. Palmer and J. M. Felsing, A Practical Guide to Feature-Driven Development, 2002.

[23] P. Coad, E. LeFebvre, J. De Luca, Java Modeling In Color With UML: Enterprise Components and Process: Prentice Hall, 2000.

[24] R. Baskerville, L. Levine, J. Pries-Heje, B. Ramesh, S. Slaughter,"How Internet companies negotiate quality," IEEE Computer, Vol. 34, pp. 51-57, 2001.

[25] R. Baskerville, J. Pries-Heje, "Racing the E- bomb: How the Internet is redefining information systems development methodology", In Realigning research and practice in IS development, B. Fitzgerald, N. Russo, and J. DeGross, Eds. New York: Kluwer, 2001, pp. 49-68.

[26] M. A. Cusumano, D. B. Yoffie,"Software development on Internet time," IEEE Computer, Vol.32, pp. 60-69, 1999.

[27] M. A. Cusumano and R. W. Selby, "How Microsoft builds software," Communications of the ACM, vol.40, pp. 53-61, 1997.

[28] D. P. Truex, R. Baskerville, H. Klein,"Growing systems in emergent organizations," Communications of the ACM, Vol. 42, pp. 117-123,1999.

[29] R. Baskerville, J. Travis, D. P. Truex,"Systems without method: The impact of new technologies on information systems development projects," In Transactions on the impact of computer supported technologies in information systems development, K. E. Kendall, K. Lyytinen, and J. I. DeGross, Eds. Amsterdam: Elsevier Science Publications, 1992, pp. 241-260.

[30] D. P. Truex, R. Baskerville, J. Travis, "Amethodological systems development: The deferred meaning of systems development methods," Accounting, Management and Information Technology, Vol. 10, pp. 53-79, 2001.

[31] A. Hunt, Thomas, D.,"The Pragmatic Programmer: Addison Wesley, 2000.

[32] K. Schwaber,"Scrum Development Process," presented at OOPSLA'95 Workshop on Business Object Design and Implementation, 1995.

[33] K. Schwaber, M. Beedle, Agile Software Development With Scrum. Upper Saddle River, NJ: Prentice-Hall, 2002.

[34] J. Iivari, R. Hirscheim,"Analyzing information systems development: A comparison and analysis of eight IS development approaches," Information Systems, Vol. 21, pp. 551-575, 1996.

[35] T. W. Olle, H. G. Sol, and A. Verrijn-Stuart, Information systems design methodologies: A comparative review. Amsterdam: North-Holland,1982.