A Method for the Selection of Software Development Life Cycle Models using Analytic Hierarchy Process

Mumtaz Ahmad Khan
Electrical Engineering Section, Faculty of Engineering and Technology, Jamia Millia Islamia (A Central University), New Delhi-110025, India
E-mail: makhan1969@gmail.com

Azra Parveen
M. Tech. Scholar, Department of Electronics and Communication Engineering, Indira Gandhi Delhi Technical University for Women (IGDTUW), Kashmere Gate, Delhi-110006, India
E-mail: parveenazra999@gmail.com

Mohd Sadiq
Software Engineering Laboratory, Computer Engineering Section, Faculty of Engineering and Technology, Jamia Millia Islamia (A Central University), New Delhi-110025, India. E-mail:sadiq.jmi@gmail.com

***Abstract*-- The success rate of software system depends upon the type of software development life cycle (SDLC) models that we employ at the time of software development process. In literature, we have identified different types of SDLC models like, agile methods, rapid application development method, and traditional methods; and choosing one of them is not an easy task according to need/criteria of the software projects. Therefore, in order to address this issue, we present a method for the selection of SDLC models using analytic hierarchy process (AHP). Finally, the utilization of the proposed approach is demonstrated with the help of an example**.

Keywords: SDLC, Decision Making Process, AHP.

I. INTRODUCTION

A software development life cycle (SDLC) models represents a process for developing an information systems [13, 20]. It includes different phases like planning, analysis, design, and implementation. On the basis of our literature review [7, 13, 20], we divide the SDLC models into three parts, i.e., heavyweight development methodology, lightweight development methodology and hybrid development methodology. Heavyweight development methodology is also known as traditional methodology and it includes waterfall model, iterative model, prototype model, spiral model, and rational unified process (RUP). Lightweight process or agile process models accentuate on agility for software development process [16]. According to the Oxford dictionary, agility means the ability to think and draw conclusion quickly. Agile process includes Extreme Programming (XP) [10, 11, 12], Dynamic System Development Methodology (DSDM) [6, 10], Adaptive Software Development (ASD) [9] Scrum [31, 32], Crystal [4, 5], and Feature-Driven Development (FDD) [22, 27] etc. In 2001, 17 software developers published the manifesto for agile software development and it is based on the following twelve principles [16, 30]:

1. Satisfy customer through early and continuous delivery of software.
2. Welcome changing requirements even in the late development phase.
3. Deliver first increment within couple of weeks and complete software within couple of months.
4. Business stakeholders and developers must work together daily throughout the project.
5. Build project according to the interested stakeholders.
6. There should be a face-to-face meeting among all the stakeholders.
7. Working software is the primary measure of the progress.
8. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Trust and respect must be maintained among all the agile members.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective.

Hybrid methodology is a combination of traditional and agile methodologies and it includes rapid application development (RAD). In 2012, Hicdurmaz [13] present a fuzzy multi criteria decision making approach to software life cycle model selection. In 2011, Geambasu et al. [7] identify the influence factors for the choice of a software development methodology. In 2008, Qureshi and Hussain [16] proposed an adaptive process model which is an extension of XP model. In a similar study,

in 2008, Qureshi and Hussain [15] proposed a model for software component based development (CBD) process model. According to the Standish report 2009 [29], projects success rate shows that 32% of all projects succeeded, 44% were challenged and 24% failed. These parameters show that the selection of appropriate methodology plays an importing role in developing software. Therefore, it motivates us to develop a methodology for the selection of SDLC. In this paper, we first identify the various criteria's of SDLC and then apply the analytic hierarchy process (AHP) for the selection of SDLC model. There are various applications of AHP, for example, in [18, 19] we have employed AHP for the prioritization of requirements.

The rest of the paper is organized as follows: Section II and section III presents the brief introduction about SDLC and AHP respectively. Proposed method for the selection of SDLC model is given in section IV. In section V, we present the case study; and finally we conclude the paper in section VI.

II. SOFTWARE DEVELOPMENT LIFE CYCLE MODELS

In this section, we briefly introduced the objective of the following SDLC model: heavyweight development methodology, lightweight development methodology and hybrid development methodology. An SDLC model is a sequence of processes that an organization/ software development company employs to conceive, design, and commercialize a software product [2]. SDLC can be seen as consisting of nine phases: (1) project inception, (2) requirements specification, (3) design, (4) coding, (5) unit test, (6) integration test, (7) system test, (8) acceptance test, and system in use [33].

A. Heavyweight Development Methodology (HDM)

The traditional or heavyweight development methodology (HDM) is based on a sequential series of steps like (i) requirements elicitation/ definition, (ii) solution build, (iii) testing and deployment. There are several HDM like waterfall model, prototyping model, spiral model etc [2, 33]. Waterfall model is a conventional, linear, sequential or traditional software life cycle model. In this model, project is divided into sequential phases. We cannot change/update any requirements based upon customer requirements. Software prototyping is the development approach of activities during software development, the creation of prototypes, i.e., incomplete versions of the software being developed. The spiral model is a software development process combining elements of both design and prototyping in stages. The focus of this model is on the risk assessment and minimizing project risk by breaking the project into smaller segments.

B. Lightweight Development Methodology (LDM)

Lightweight methodologies are also referred to as agile methodologies or agile software development methodologies. These methodologies are characterized by the following attributes [3, 20]: adaptive, incremental, cooperative and straightforward. On the basis of our literature review , we have identified the following agile methods, i.e., Adaptive Software Development [9], Agile Modeling [26], Crystal Family [4, 5], Dynamic System Development Method [6, 8], Extreme Programming [10, 11, 12], Feature Driven Development [22, 27], Internet Speed Development [14, 23], Pragmatic Programming [34], and Scrum [31,32].

a. Adaptive Software Development (ASD)

ASD provides a framework to guide that how to prevent projects from falling into chaos. RADical software development [9, 35] is an ancestor of ASD. ASD offers solutions for the development of large and complex system and encourage incremental and iterative development with the constant prototyping. Qureshi and Hussain [16] proposed an adaptive process model for agile development; and it is modified approach of the extreme programming (XP). The main phases of adaptive process models includes: communication, and planning, analysis, design and development, testing and deployment.

b. Agile Modeling (AM)

Agile modeling (AM) was introduced by Scott W. Ambler in 2002 for modeling activities [26]. It keeps the amount of models and documentation as low as possible [20]. Four elements of agile modeling are similar to XP, i.e., Communication, simplicity, feedback and courage. The idea behind agile modeling is to produce sufficiently advanced models according to the need of the customer.

c. Pragmatic Programming (PP)

It introduces a set of programming best practices. It is a collection of short tips (there are total 70 of them) that focus on day- to -day problems. It does not have process, phase, or work products. It covers most programming practicalities. The philosophy of PP is universal and can be applied to any software development phase [20, 34].

d. Scrum

Scrum is an iterative and incremental agile method for managing software projects. It is an empirical approach based on the following: adaptability, flexibility, and productivity. In scrum, projects are divided into sprints. Each sprint has one week to four week duration. The sprints are of fixed in duration and never extended. The objective of each sprint is to produce usable product with an added increment of function [31, 32].

e. Crystal Family

Crystal method is one of the most popular agile methods and it was developed by Alistair Cockburn in 1990's. This method was developed to address the frequently changing project environment and characteristics. It includes the number of different methods from which to select the most suitable one for the project. The benefits of crystals family includes: (a) improve communication and collaboration throughout the project team (b) improves system performance (c) deliver faster results and enhances development process. In this methodology, different methods are assigned different color on the basis of their agility. The order of methods from most agile to least is: (i) crystal clear, (ii) crystal yellow, (iii) crystal orange, and (iv) crystal red. Crystal methods are open for any development practices, tools, and work products [4, 5, 20].

f. Dynamic System Development Method (DSDM)

DSDM was developed in U.K. in the mid 1990. It is similar in many ways to scrum and XP, but it is used in those projects where the time requirements are fixed. In traditional development methodologies, functionality is fixed and the time and resources are variable, but in case of DSDM, time is fixed and functionality varies according to the need of stakeholders. There are nine principles of DSDM like (1) active user involvement; (2) team must be empowered to take the decision; (3) frequently release of the product; (4) iterative development; (5) reversible changes during the development; (6) requirements are initially defined at high level; (7) meeting the business need is more important; (8) integrating testing throughout the life cycle (9) collaboration and cooperation are essential [6, 8].

g. Extreme Programming (XP)

XP is used to improve the software quality and supports changing customer requirements. This methodology stresses on team work. Managers, developers, and customers are all equal parterres in collaborative team. XP is based on simplicity, communication, feedback, and courage. XP is popular because of its management activities, for example, (i) it involves those activities that would be used to decide what should be done in the next phase and also (ii) predicts when the project will be done [10, 11, 12].

h. Feature Driven Development (FDD)

FDD emphasize on feature aspect of a project. In 1997, this methodology was first applied on a 15 month, 50-person project for a large Singapore bank and the same method was immediately followed by a second, 18-month long 250-person project. There are five main activities in FDD that are performed iteratively: (i) develop an overall model (ii) build a features list (iii) plan by features (iv) design by features (v) build by features [22, 27].

i. Internet Speed Development (ISD)

It is least known approach in agile literature. It is used in those situations where software needs to be released fast, and requires short development cycles. This framework consists of time drivers, quality dependencies, and process adjustments [14, 23].

C. Hybrid Development Methodology (HYDM)

HDM and LDM have their own advantages and disadvantages. For example, agile methods sometimes lack some of the characteristics of traditional methods or HDM that are crucial for success. Therefore, a hybrid solution may be the right answer for those projects which require the combination of both the methodologies. Rapid Application Development is an HYDM because it combines the elements from both traditional and agile methodologies [7]. In 2009, Cho [37] proposed a hybrid software development method for large-scale projects by combining rational unified process (RUP) with scrum.

III. ANALYTIC HIERARCHY PROCESS

In 1972, T. L. Saaty proposed the analytic hierarchy process [28]. It is a multi-criteria decision (MCDM) making method. AHP helps decision maker facing a complex problem with multiple conflicting and subjective criteria [1, 28]. This process permits the hierarchical structure of the criteria or sub-criteria when allocating a weight. AHP has been widely used in various fields like banks, manufacturing systems, software evaluation, requirements prioritization [17, 18, 19], evaluation of web site performance etc. AHP involves following steps: (a) problem definition (b) pair-wise comparisons (c) compute the eigenvector of

the relative importance of the criteria (d) check consistency. Once we have identified the criteria or sub-criteria according to the need of the problem or **problem definition**, then the next step is to express the decision makers opinion on only two alternatives than simultaneously on all the alternatives. On the basis of the pair wise comparison with all the alternatives, we construct the **pair-wise comparison matrix** on the basis of the following rating scale **(Judgment scale)**. Table 1 presents the rating scale proposed by saaty.

Table: 1 the Saaty rating scale

Intensity of importance	**Definition**
1	Equal importance
3	Somewhat more importance
5	Much more important
7	Very much important
9	Absolutely more important
2,4,6,8	Intermediates values (when compromise is needed)

There are several methods or algorithm for the calculation of eigenvector. In this paper, we adopt the following algorithm:

Algorithm:

Step 1: Multiplying together the entries in each row of the matrix and then take the n^{th} root of the product.
Step 2: Compute the sum of n^{th} root and store the result in SUM.
Step 3: The value of SUM would be used to normalize the product values and the resultant would be the eigenvector

Saaty argues that a Consistency Ratio (CR) >0.1 indicates that the judgment are at the limit of consistency, where as CR=0.9 would mean that the pair wise judgment are random and are completely untrustworthy [1, 28].

IV. PROPOSED METHOD

This section presents a method for the selection of SDLC models using AHP. The proposed method is presented simply in the following:
(i) Identify the criteria
(ii) Construct the hierarchical structure of SDLC model
(iii) Construct the decision matrix
(iv) Calculate the ranking values
(v) Selection of a model

(i) Identify the criteria

Before the selection of any SDLC model, requirements analyst should identify the criteria's for the selection of an SDLC model. On the basis of our literature review, we have identified the following factors which influence the decision of choosing a software development methodology:
(a) Clarity of the requirements
(b) Cost and development time
(c) Incorporation of requirements change
(d) System complexity
(e) Communication among different stakeholders
(f) Size of development team etc.

(ii) Construct the hierarchical structure of SDLC model

As the SDLC selection decision requires a systematic approach to help integrate different attributes or criteria into software project development. Therefore, it is essential to break down the problem into more manageable sub-problems. As illustrated in Fig.1, the problem studied here has three level of hierarchy. The first level, i.e., the overall objective, is the selection of an SDLC model. Level two contains three different SDLC model, and at level three decision criteria is given.

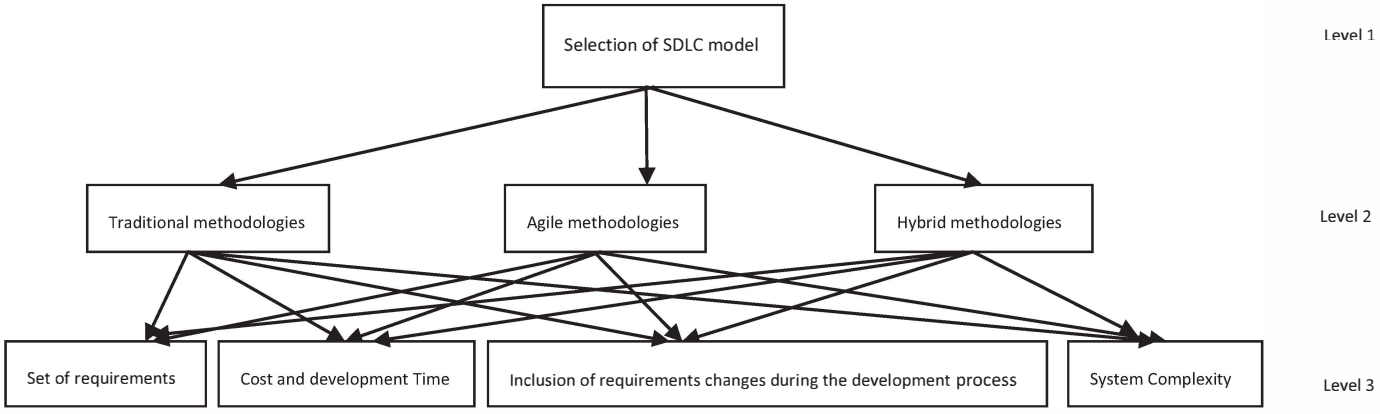

Fig.1 Hierarchical Structure of the SDLC selection problem

(iii) Construct the Decision Matrix

We will create the decision matrix using AHP method [1, 17, 18]. Detailed description for the construction of decision matrix is given in section V.

(iv) Calculate the ranking values

Ranking values will be obtained after computing the eigenvector values from the pair wise comparison matrix [1, 17, 18].

(v) Selection of a model

Construct the binary search tree (BST) of the ranking values that we have obtained in previous step. Apply in-order tree traversal technique on BST and as a result, we will get the prioritized list of SDLC models. The model which has highest priority will be selected for the development of the project.

V. CASE STUDY

This section presents a case study of our work. Before the starting of the projects, it is indispensible to select the best SDLC model according to the need of our project. There are various SDLC models which are available in the literature. Each model is designed according to certain requirements; and each model has also some advantages and disadvantages. For example, Rational Unified Process (RUP) requires the complete set of requirements in the beginning of the project, while on the other hand side, XP applies agile concepts and does not require complete set of requirements in the beginning of the project. In this paper, we have considered the project, developed by our students, entitled "Computer Graphics Package". Therefore, the objective of the proposed method is the selection of SDLC model. We have identified the following criteria for the selection of SDLC model *(Step 1st)*: complete set of requirements (CSR), cost and development time (CDT), inclusion of requirements change during the development process (IRCDT), and system complexity (SC). The hierarchical structure of the SDLC selection problem is given in Fig. 1 (Step 2nd). For the 3rd step, we have defined the initial matrix for the pair wise comparison. In this matrix, the principal diagonal matrix contains entries of 1 because each factor is important as itself.

Table 2 Initial Matrix

Criteria	CSR	IRCDT	CDT	SC
CSR	1			
IRCDT		1		
CDT			1	
SC				1

To make the pair wise comparison among all the criteria, we decide that IRCDT is more important than CSR. In the next matrix, i.e., Table 3, that is rated as 3 in the cell IRCDT and CSR; and 1/3 in CSR and IRCDT. We also decide that CSR is more important than CDT. Therefore, in Table 4, we put 5 in CSR and CDT; and 1/5 in CDT and CSR. In a similar way we complete the matrix, that we call the *"Overall Preference Matrix (OPM)"*.

Table 3

Criteria	CSR	IRCDT	CDT	SC
CSR	1	1/3		
IRCDT	3	1		
CDT			1	
SC				1

Table 4 Overall Preference Matrix

Criteria	CSR	IRCDT	CDT	SC
CSR	1	1/3	5	1
IRCDT	3	1	5	1
CDT	1/5	1/5	1	1/5
SC	1	1	5	1

The eigenvector or relative value vector (RVV) corresponding to each criterion is calculated by the algorithm, given in section III 9 (Step 4th) . Therefore, as a result, we have identified the following values: (0.232, 0.402, 0.061, 0.305). These four values correspond to the relative value of CSR, IRCDT, CDT, and SC. The value 0.402 means that IRCDT is an important criterion. 0.305 shows that SC is also important parameter for the selection of SDLC. The remaining two data represent that CDT and CSR are least considerable parameters.

After this we evaluate different SDLC models on the basis of the given parameters, i.e., CSR, IRCDT, CDT, and SC. Table 5, Table 6, Table 7, and Table 8 are according to the CSR, IRCDT, CDT, and SC; and it ranks the three SDLC models, i.e., Heavyweight Development Methodology (HDM), Lightweight Development Methodology (LDM), and Hybrid Development Methodology (HYDM).

Table 5 According to CSR

Criteria	HDM	LDM	HYDM
HDM	1	5	9
LDM	1/5	1	3
HYDM	1/9	1/3	1

Table 6 According to IRCDT

Criteria	HDM	LDM	HYDM
HDM	1	1	5
LDM	1	1	3
HYDM	1/5	1/3	1

Table 7 According to CDT

Criteria	HDM	LDM	HYDM
HDM	1	1/3	1/9
LDM	3	1	1/3
HYDM	9	3	1

Table 8 According to SC

Criteria	HDM	LDM	HYDM
HDM	1	1/9	1/5
LDM	9	1	2
HYDM	5	1/2	1

As a result, we have identified the following values: (0.232, 0.402, 0.061, 0.305). On the basis of our analysis, we identify that: HDM is far important then LDM and HYDM in terms of CSR. The value 0.402 indicates that IRCDT is an important criterion for the selection of SDLC model (Step 5th). On the basis of SC value, proposed algorithm select or motivate to use agile methods for the development of projects. The projects developed by students are less complex and less experienced students are involved in the development process. The agile methods are employed when project are developed in an adaptive, incremental, cooperative and straightforward way. The reasons for the selection of agile method are given as follow:

(a) It is based on short life cycle.
(b) It allows programmer to build solutions more quickly etc.

On the basis of our analysis, we have identified the following limitations in agile methods [20, 36]:

(i) AM and PP do not support for project management.
(ii) AM and ISD, Scrum, and PP do not accentuate the process through which the software development proceeds.
(iii) Agile process models are not suitable to develop all types of software; and it is not suitable for large team and less experienced developers.

VI. CONCLUSION

This paper presents a method for the selection of SDLC models using AHP. Proposed method is a five step process, namely, (i) identify the criteria, (ii) construct the hierarchical structure of SDLC model, (iii) construct the decision matrix, (iv) calculate the ranking values, and (v) the selection of a model. Proposed method selects the agile methods for the development of the project. On the basis of our analysis, we identify that there is a need to improve the agile methods by intertwining of decision making approaches for the selection and prioritization of requirements. Future research agenda includes the following:

1. To improve the analysis phase of adaptive process model for agile development by applying TOPSIS method.
2. To propose a fuzzy decision making approach or the selection of SDLC model.
3. To propose a hybrid approach of SDLC model.
4. To propose a method for the selection of SDLC models using hybrid techniques like fuzzy AHP and fuzzy ANP.

References

1. A. Ishizaka and A. Labib, "Review of the Main Development in the Analytic Hierarchy Process", Experts Systems with Applications, Vol.38, No.11, pp. 14336-14345, 2011.
2. B.W. Boehm, "A Spiral Model of Software Development and Enhancement", IEEE Computer, Vol. 21, pp.61-72, 1988.
3. Cockburn, Agile Software Development, Boston: Addison-Wesley, 2002.
4. A. Cockburn, Surviving Object Oriented Projects: A Managers Guide, Vol.5 Addison Wesley Longman, 1998.
5. A. Cockburn, Writing Effective Use Cases, the Crystal Collection for Software Professionals: Addison-Wesley Professional, 2000.
6. DSDM Consortium, Dynamic System Development Method, Version 3, Ashford, Eng: DSDM Consortium, 1997.
7. Geambasu and I. Jianu, "Influence factors for the Choice of a Software Development Methodology", Accounting and management Information Systems, Vol. 10, No.4, pp. 479-494, 2011.
8. J. Stapleton, Dynamic Systems Developments Method – The Method in Practice: Addison Wesley, 1997.
9. J. A. High Smith, Adaptive Software Development: A collaborative Approach to Managing Complex Systems, New York, NY: Dorest, House Publishing, 2000.
10. K. Beck, Extreme Programming Explained. Embrace Change, 2000.
11. K. Beck, Extreme Programming Explained. Reading Mass.: Addison-Wesley, 1999.
12. K. Beck, Embracing Change with Extreme Programming", IEEE Computer, Vol. 32, pp. 70-77.
13. M. Hicdurmaz, "A Fuzzy Multi Criteria Decision Making Approach to Software Life Cycle Model Selection", 38th IEEE EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 384-391, 2012.

14. M. A. Cusumno and D. B. Yoffie, "Software Development on Internet Time", IEEE Computer, vol. 32, pp.60-69, 1999.
15. M. R. J. Qureshi and S. A. Hussain, "A Reusable Software Component based Development Process", Advances in Engineering Software, Elsevier, Vol. 39, pp.88-94, 2008.
16. M. R. J. Qureshi and S. A. Hussain, "An Adaptive Software Development Process Model", Advances in Engineering Software, Elsevier, Vol.39, pp.654-658, 2008.
17. Mohd Sadiq, S K Jain, " A Fuzzy Based Approach for Requirements Prioritization in Goal Oriented Requirements Elicitation Process", 25th International Conference on Software Engineering and Knowledge Engineering, Boston, USA, June 27-June 29, 2013.
18. Mohd. Sadiq, Jawed Ahmad, Mohammad Asim, Aslam Qureshi , R. Suman, " More on Elicitation of Software requirements and prioritization using AHP", IEEE International Conference on Data Storage and Data Engineering pp 232-236, Bangalore, India, 2010
19. Mohd. Sadiq, Shabina Ghafir, Mohd. Shahid, "An Approach for Eliciting Software Requirements and its Prioritization using Analytic Hierarchy Process", IEEE International Conference on Advances in Recent Technologies in Communication and Computing, pp.790-795 Kerala, India, 2009.
20. P. Abrahamsson, J. Warsta, M T Siponen, and Jussi Ronkainen, "New Directions on Agile Methods: A Comparative Analysis", IEEE International Conference on Software Engineering, U.S.A., 2003.
21. P. Abrahamsson, O salo, J. RonKainen, and J. Warsta, Agile software development methods: Review and Analaysis, Espoo, Finland: Technical research Center of Finland, VTT Publication 478.
22. P. Coad, E. Lefebvre, and J. DeLuca , Java Modeling in Color with UML: Enterprise Components and Process: Prentice Hall, 2000.
23. R. Baskerville and J. Priess- Heje," Racing the E-bomb: How the Internet is Redefining Information Systems Developments methodology," in Realigning research and practice in IS Development, B. Fitzrearald, N. Rosso, and J. Degross, Eds. New York: Kulewer, 2001, pp.49-68.
24. R. Baskerville, L. Levine, J Pries-Heje, B. Ramesh, and S. Slaughter, "How Internet Companies Negotiate Quality," IEEE Computer,Vol.34, pp.51-57, 2001.
25. Review and Analysis. Espoo, Finland: Technical Research Centre of Finland, VTT Publications 478,
26. S. Ambler, Agile Modeling: Effective Practices for Extreme Programming and the Unified Process. York: John Wiley & Sons, Inc. New York,2002
27. S. R Palmer and J.M Felsing, "A Practical Guide to Feature-Driven Development", 2002.
28. Saaty T, "The Analytic Hierarchy Process", New York: McGraw-Hill, 1980.
29. The Standish Group International, CHAOS Summary 2009.
30. Agile Alliance, Principles behind the Agile Manifesto: http:// agilemanifesto.org / principles.html.
31. K Schwaber, "Scrum Development Process", Presented at OOPSLA'95 Workshop on Business Object Design and Implementation, 1995.
32. K. Schwaber and M. Beedle, "Agile Software Development with Scrum, Upper saddle River, NJ: Prentice Hall, 2002. ISBN: 0130676349.
33. I. Sommerville, "Software Engineering", Fifth Edition, New York: Addison Wesley, 1996.
34. A Hunt, Thomas D, "The Pragmatic Programmer", Addison Wesley, 2000.
35. S. Bayer and J. Highsmith, " Radical Software Development", American programmer, Vol. 7, pp. 35-42, 1994.
36. J. Highsmith, A. Cockburn, "Agile Software Development: The People Factor", IEEE Computer, Vol. 34, pp. 131-133, 2001.
37. Juyun Cho, "A Hybrid Software Development Method for Large-Scale Projects: Rational Unified Process With Scrum", Vol. 10, No.2, pp. 340-348, Issue in Information Systems, 2009