Superiority of Agile over Software Models

Javed Ali

Abstract: ***The Software Development Life Cycle (SDLC) is a structure imposed on the development of software product and also known as Software Development Processes. It is often considered as a subset of a system development life cycle. There are numerous SDLC models widely used for developing software. The SDLC model gives a theoretical guide line regarding development of the software. The every SDLC has its own benefits and drawbacks according to that we decide which models should be implemented under which conditions. The concept of system lifecycle models came into existence that emphasized on the need to follow some structured approach towards building new or improved system. In this paper, the comparative study made on standard life cycle models namely rapid application development, Agile Development Model, V-shaped model, spiral model, incremental model and waterfall model, prototype.***

Keywords: Software Development, Life Cycle, Model, Agile Development, Extreme Programming, Dynamic, Adaptive.

I. INTRODUCTION

The Software development life cycle (SDLC) is a method by which the software can be developed in a systematic manner and which will increase the probability of completing the software project within the time deadline and maintaining the quality of the software product as per the standard. The System Development Life Cycle framework provides a sequence of activities for system designers and developers to follow for developing software. It is often considered as a subset of system development life cycle [1] [2]. Any software development process is divided into several logical stages that allow a software development company to organize its work efficiently in order to build a software product of the required functionality within a specific time frame and budget. A Software Development Life Cycle Model (SDLC) is a set of activities together with an ordering relationship between activities performed in a manner that satisfies the ordering relationship that will produce desired product [3] [4] [5].

The SDLC Model is an abstract representation of a development process. In a software development effort the goal is to produce high quality software. The development process is, therefore, the sequence of activities that will produce such software. A software development life cycle model is broken down into distinct activities and specifies how these activities are organized in the entire software development effort [6] [7]. In response to traditional approaches to software development, new lightweight methodologies have appeared. A high percentage of software development efforts have no process and might best be described as a chaotic "code and fix" activity.

The software planning management provides the technical way starting how to implement the software methods that include Communication,

Requirement Analysis, Analysis and Design modeling, Program construction, Testing and support [8]. Software development life cycle SDLC is the systematic approach to complete the software development process within the time and maintain quality of the software. System development life cycle provides the set of activities to be carried out during the system development and it is often called that software development life cycle. Software development is divided into set of activities that allow any software development company to control the software product easily [9] [10].

II. RELATED WORKS

2.1. Water Fall Model (WFM)

The Waterfall Model (WFM) known as a linear sequential life cycle model. It is the first Process Model to be introduced, very simple to understand and expenditure. In WFM, every stage starts when the previous one has completed and there is no coinciding in the stages.

All stages in WFM have its own deliverables. It is expectable and values demanding software planning architecture. The software development projects where requirements are clear and detailed planning can be easily draft for the entire project.

The basic principles of water fall model are: Project is divided into sequential phases with some connexion and wade back acceptable between phases and tight control is maintained over the life of the project via extensive written documentation, formal reviews and approval/sign off by the user and information technology management occurring at the end of most phases before beginning the next phase.

The WFM has major advantage; every stage has well defined deliverable or innovatory [1] [11]. It has also disadvantages, small changes or errors that arise in the completed software may cause a lot of problems and if the design phase has gone wrong things can get very complicated in the implementation stage.

Revised Version Manuscript Received on December 08, 2016.

Javed Ali, Assistant Professor, College of Computing Informatics, Madinah Munawarah Branch, Saudi Electronic University, KSA.

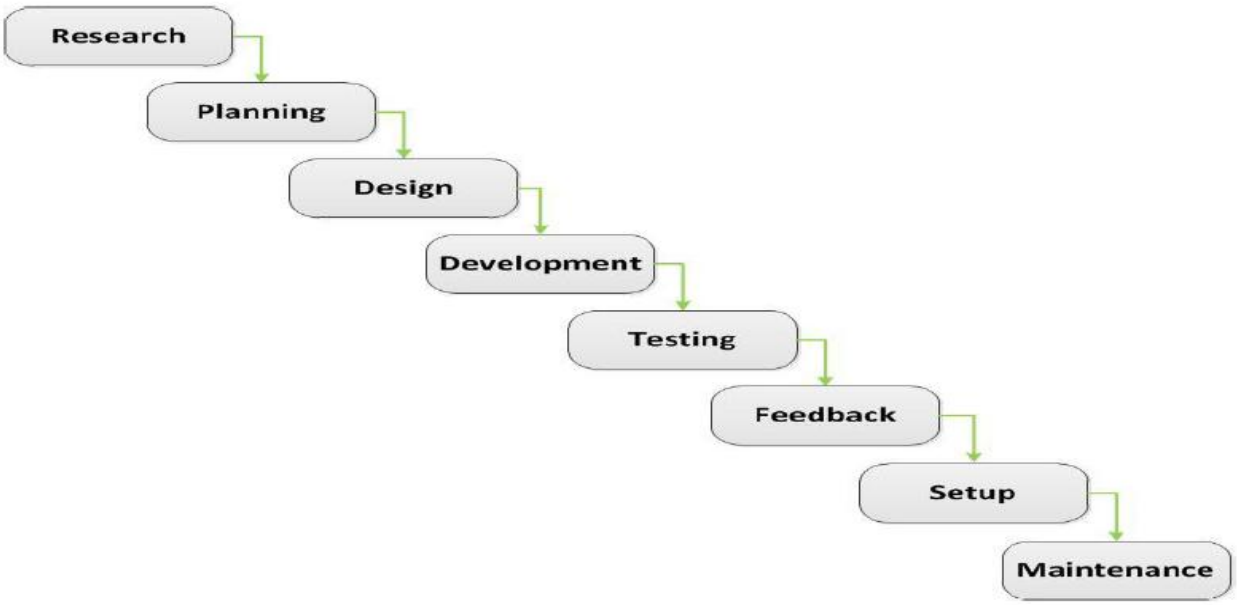

Figure 1: The Water Fall Model Approach

2.2. V- Shaped Model (VSM)

The V-Shaped life cycle is a chronological lane of implementation of processes. It works same as waterfall model. Each phase is compulsory to complete before the next phase begins. Testing is highlighted in this model more than the waterfall model. The testing actions are developed early in the life cycle before any coding is done, during each of the phase's previous implementation. Requirements start the life cycle model as the waterfall model. Earlier than development is started, a system test plan is created. In this model testing is spotlighted on meeting the functionality specified in requirements gathering. The main design phase spotlight on system architecture and design. The combine test plan is created in this phase in order to test the pieces of the software systems capacity to work jointly [13]. Though, the low-level design phase lies where the real software components are designed, and unit tests are created in this phase. The accomplishment phase is, again, where all coding is generated. After coding is complete, the way of execution continues up the right side of the V where the test plans developed earlier can use.

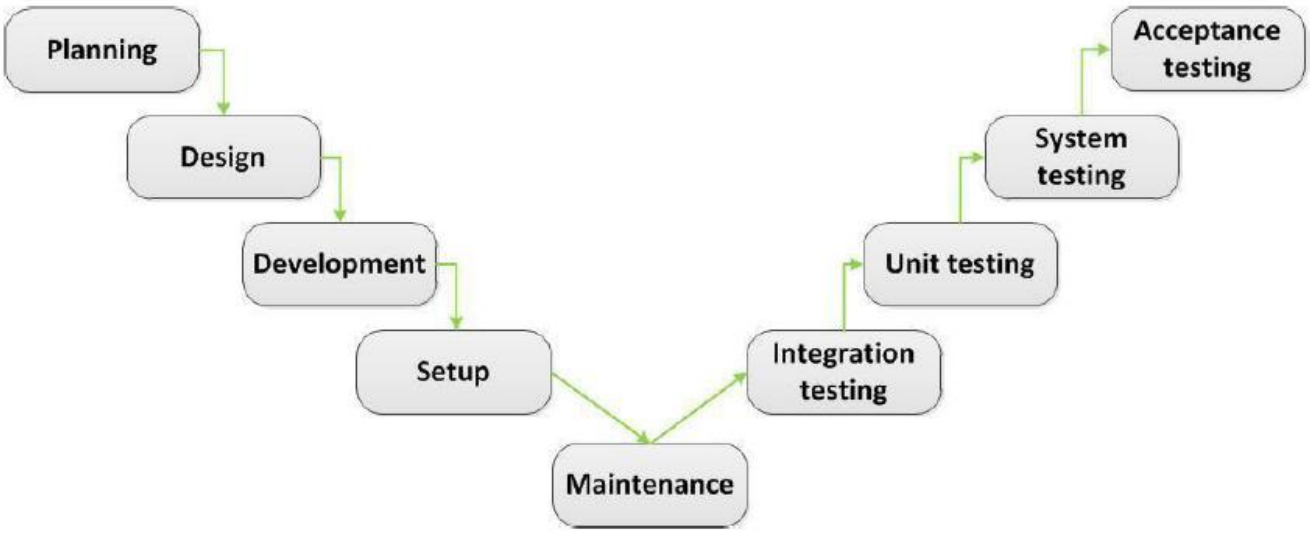

Figure 2: The V- Shaped Model Approach

2.3. Spiral Model (SM)

The spiral model is a software development process combining elements of both design and prototyping-in-stages. The model divided in four phases: Planning, Risk Analysis, Engineering and Development. A software project frequently passes through these phases in iterations (called Spirals in this model). At the initial spiral, starting with the planning, requirements are gathered and risk is considered. Each consequent spiral builds on the initial spiral. Requirements are collect during the planning phase. In the risk analysis phase, a process is going on to identify risk and their alternate solutions [4] [14]. A prototype is produced at the end of the risk analysis. The evaluation phase permits the customer to assess the output of the project to date before the project goes to the next spiral.

The spiral model has four basic principles. The first is focus on risk assessment and on minimizing project risk by breaking a project into smaller segments and providing more ease-of-change during the development process,

as well as providing the opportunity to evaluate risks and weigh consideration of project continuation throughout the life cycle. The second one, each cycle involves a progression through the same sequence of steps, for each part of the product and for each of its levels of elaboration, from an overall concept-of-operation document down to the coding of each individual program. In third one, each trip around the spiral traverses four basic quadrants: (1) determine objectives, alternatives, and constraints of the iteration; (2) evaluate alternatives; Identify and resolve risks; (3) develop and verify deliverables from the iteration; and (4) plan the next iteration. The fourth and last one, begin each cycle with an identification of stakeholders and their win conditions, and end each cycle with review and commitment.

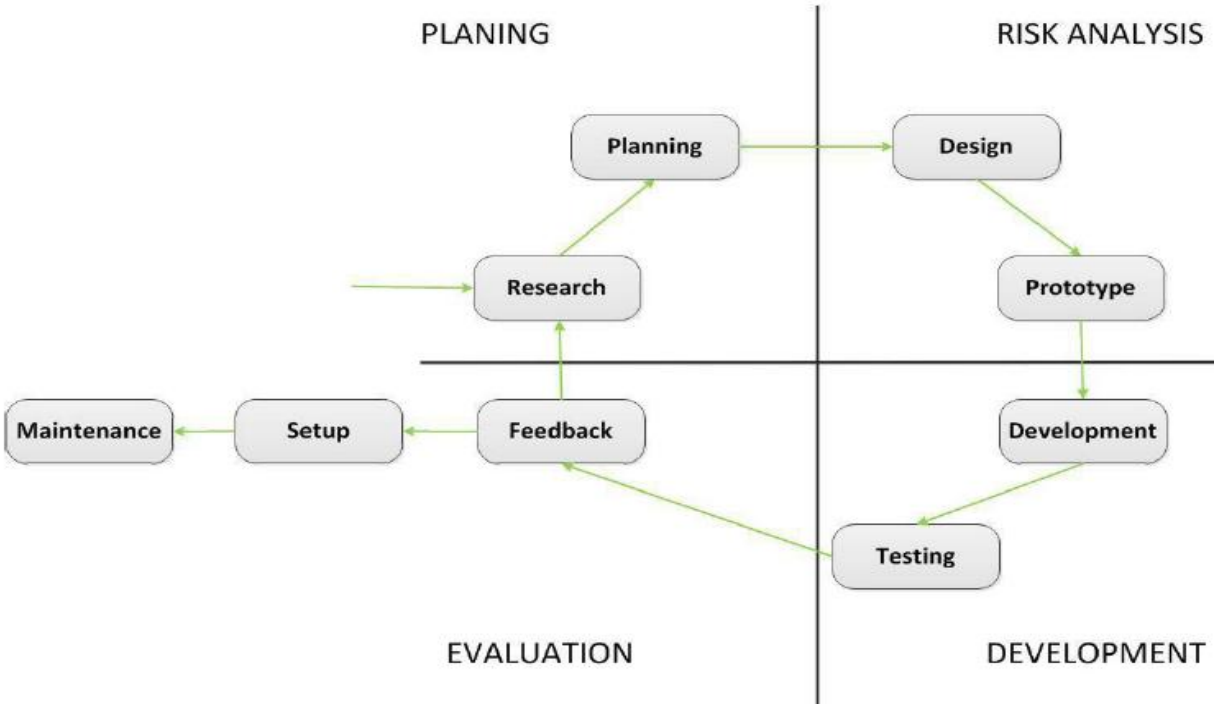

Figure 3: The Spiral Model Approa

2.4. Incremental Model (IM)

The incremental model (IM) may start with general objectives which is also defined as provisions and those requirements are implemented by following the next coming portion of the objectives, till all objectives are performed it push difficult problems to the future to prove early success and this model can be useful for the assignments where the basic software functionality is required at the beginning [15].The incremental expansion should be measured when it is perilous to develop the whole system at once. The IM is lacked in: It needs good planning and design and also needs a strong complete description of the entire system before it can be broken down and built incrementally. The total cost of IM is higher in comparison to waterfall model.

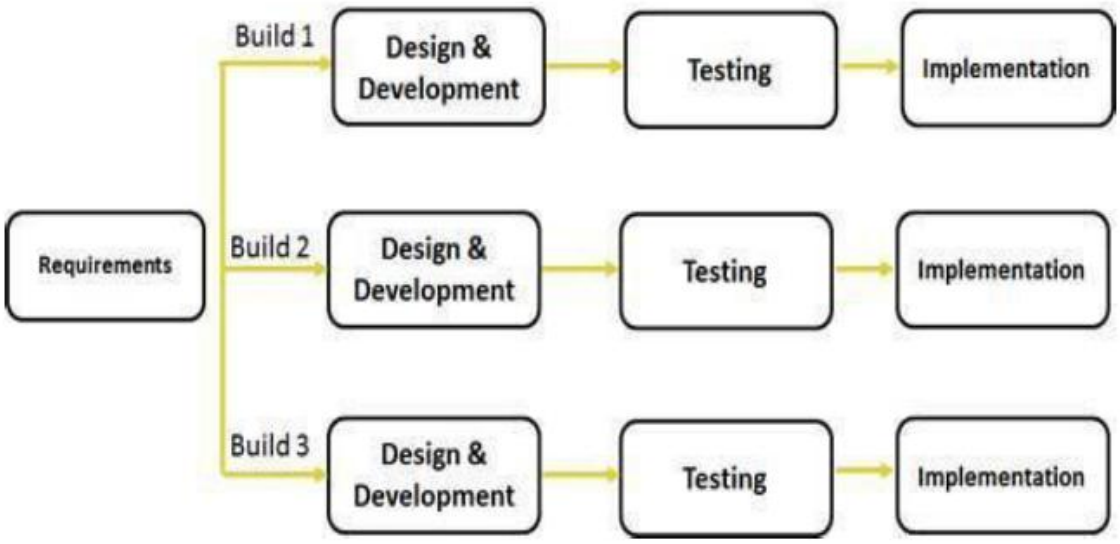

Figure 4: The Incremental Model Approach

2.5. Rapid Application Development Model (RADM)

The Rapid Application Development model (RADM) is most similar of incremental model. The components in RADM are developed in parallel. The development are time boxed, delivered and then bring together into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements. In RADM requirements are well understood and project scope is constrained, the RADM, figure 5 process enables a development team to create a "fully functional system" within very short time periods [7] [8] [17].

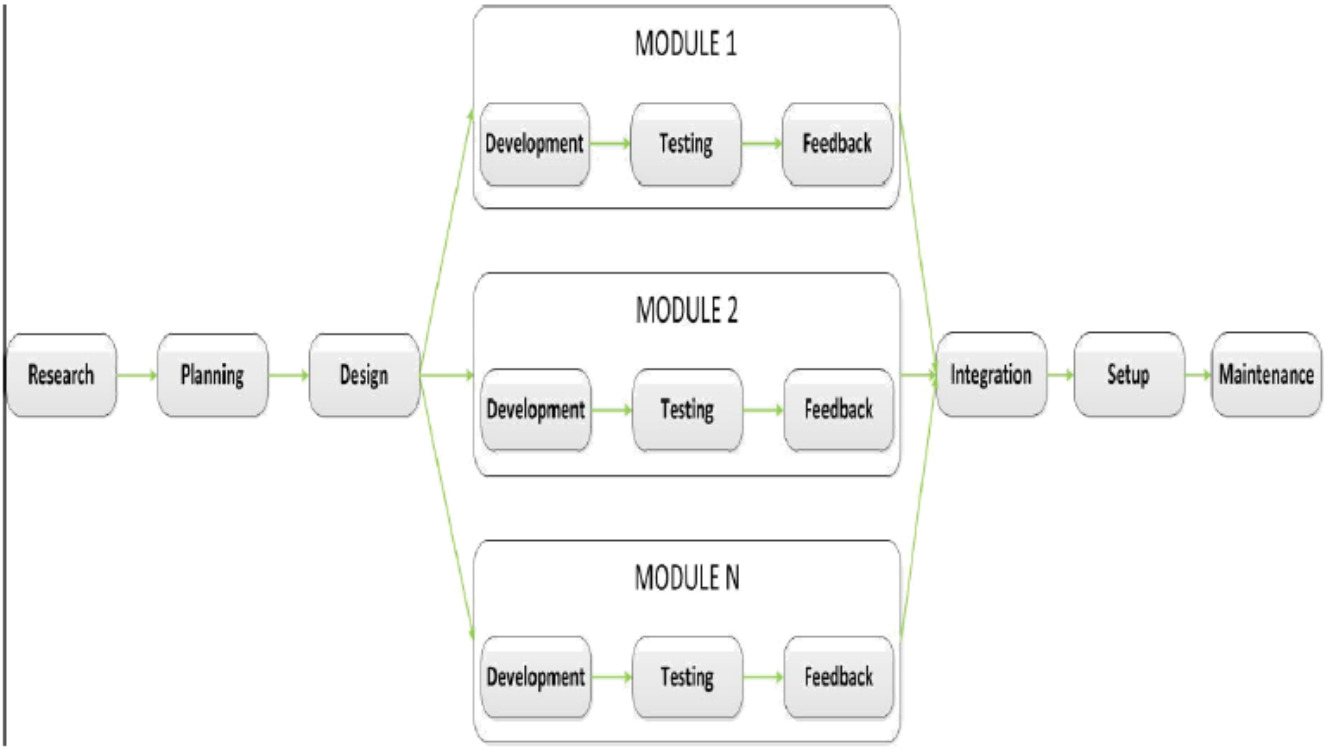

Figure 5: The Rapid Application Development Model Approach

2.6. Agile Development Process Model

The numerous companies are reluctant to evade their out-of-date methods and hurdle into agile methods. Their reluctance is the result of several problems that contain agile methods meaningfully diminish the amount of documentation and rely heavily on tacit knowledge, for mission/safety-critical projects, and it also belief that these methods are not passable for highly stable projects, a major concern that agile methods can be successful only with talented individuals who require many degrees of freedom, and that agile methods are not appropriate for very largescale projects.

2.6.1 Different Assumptions

The determination towards agile started in the nineties. It can be describe as iterative and incremental development including flexibility throughout the systems development life cycle, minimal groundwork, light and fast development cycles, people-centric development, customer relationship, and regular delivery. In 2001, seventeen experts met at Snowbird, Utah, to discuss if there was anything in common among the various agile methods (Cockburn, 2007) and they formed the Strategy for Agile Software Development (Becket al., 2001), which also uncovered what items were unrushed precious by ASDMs. Highest priority is to accomplish the customer demands early and provide continuous delivery of esteemed software [1] [2] [16]. It also appreciates the changing requirements, even if it affects the development process and the most proficient and working method of freeing information to and within a development team is face-to-face discussion. Working software is the primary measure of any progress. Agile actions endorse maintainable progress. The best styles requirements and designs appear from self-organizing teams. At continuous breaks, the team reflects on how to become more effective, then tunes and adjusts its performance accordingly. Also we have the some basic principles are given below:

- Our highest priority is to satisfy the customer through early and continuous delivery of products.
- Appreciate changing necessities, for the user modest benefit.
- Deliver working software frequently, within few days with a preference to the shorter timescale.
- Business customers and developers must work together on daily basis throughout the project.
- Projects should be developed around motivated people and they should be provided by the atmosphere and also by the support and trust to get the work completed.
- The most 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 endorse supportable enlargement. The guarantors, designers, and employers should be able to maintain a continuous step open-endedly.
- Continuous attention to technical excellence and good design enhances agility.

III. COMPARATIVE STUDY OF SOFTWARE DEVELOPMENT PROCESS MODELS (SDPM)

Table: 1. The Comparison of SDPM

Model Parameters	WFM	VSM	SM	IM	RADM	AGILE
Requirement Specification	Beginning	Beginning	Beginning	Beginning	Beginning	Frequently Change
Cost	Low	Expensive	Expensive	Low	Low	Very High
Simplicity	Simple	Intermediate	Intermediate	Intermediate	Very Simple	Complex
Maintenance	Least Glamorous	Least	Typical	Promote Maintainability	Easily Maintainability	Promote Maintainability
Overlapping Phase	No Such Phase	No	Yes	No	No	Yes
Reusability	Limited	To some Extent	Yes	Yes	Some Extent	User Case Reuse
Interface	Minimal	Minimal	Crucial	Crucial	Minimal	Model Driven
Risk Analysis	Only at Beginning	Yes	Yes	No risk Analysis	No	Yes
Flexibility	Rigid	Little Flexible	Flexible	Less Flexible	High	High Flexible
Time Frame	Long	According To project size	Long	Very Long	Short	Least Possible

IV. CONCLUSION AND FUTURE SCOPE

There are plenty of SDLC models for instance WFM, RAD, SM, IM as well as VSM employed in numerous companies relying upon the circumstances existing generally there. Each one of these various software development models possesses their unique pros and cons. In the Software Industry, the combination of most these kinds of methodologies is employed i .e. with certain customization. Within this paper we certainly have compared the distinct software development life cycle models on the basis of particular capabilities like- Requirement specifications, Risk involvement, Simplicity, Maintenance, Overlapping Phase, Cost, Reusability. on the basis of such capabilities for a specific software project one can choose which of such software development life cycle models ought to be selected for the specific project. Choosing the appropriate life cycle model is vital in a software industry as the software needs to be shipped within the precious time deadline & are likely to possess the preferred top notch. This research can make the strategy of picking the SDLC model easy& therefore will prove to be extremely effective for software industry.

REFERENCES

1. Agrawal, Ashish, Mohd Aurangzeb Atiq, and L. S. Maurya. "A Current Study on the Limitations of Agile Methods in Industry Using Secure Google Forms."Procedia Computer Science 78 (2016): 291-297.
2. Raj, Gaurav, Dheerendra Singh, and Ankur Bansal. "Analysis for security implementation in SDLC."
3. In Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference-, pp. 221-226. IEEE, 2014.
4. Sharma, Anubha, Manoj Kumar, and Sonali Agarwal. "A Complete Survey on Software Architectural Styles and Patterns." Procedia Computer Science 70 (2015): 16-28.
5. Langer, Arthur M. "System Development Life Cycle (SDLC)." In Analysis and Design of Information Systems, pp. 10-20. Springer London, 2008.
6. Montini, Denis Ávila, Danilo Douradinho Fernandes, Francisco Supino Marcondes, Paulo Marcelo Tasinaffo, Italo Santiago Vega, and Luiz Alberto Vieira Dias. "Formal Approach Use to Choose a Software Manufactoring Cell's SDLC." In 2010 Seventh International Conference on Information Technology, pp. 1304-1305. IEEE, 2010.
7. Tohidi, Hamid. "The Role of Risk Management in IT systems of organizations." Procedia Computer Science 3 (2011): 881-887.
8. Kumar, Chandan, and Dilip Kumar Yadav. "A Probabilistic Software Risk Assessment and Estimation Model for Software Projects." Procedia Computer Science 54 (2015): 353-361.
9. Molokken-Ostvold et.al, "A comparison of software project overruns - flexible versus sequential development models", Volume 31, Issue 9, Page(s): 754 – 766, IEEE CNF, Sept.2005.
10. Niazi, Mahmood, Muhammad Ali Babar, and June M. Verner. "Software Process Improvement barriers: A cross-cultural comparison." Information and software technology 52, no. 11 (2010): 1204-1216.Dybå, Tore. "An empirical investigation of the key factors for success in software process improvement." Software Engineering, IEEE Transactions on31, no. 5 (2005): 410-424.
11. Liao, Li, Yuzhong Qu, and Hareton KN Leung. "A software process ontology and its application."(2003).

12. Laura C. Rodriguez Martinez, Manuel Mora, Francisco. Alvarez, "A Descriptive/Comparative Study of the Evolution of Process Models of Software Development Life Cycles", Proceedings of the 2009 Mexican International Conference on Computer Science IEEE Computer Society Washington, DC, USA, 2009.
13. Sharma, B.; Sharma. N, "Software Process Improvement: A Comparative Analysis of SPI models", Emerging Trends in Engineering and Technology (ICETET), 2009 2nd International Conference on,16-18, 2009, pp. 1019- 1024.
14. M. Davis, H. Bersoff, E. R. Comer, "A Strategy for Comparing Alternative Software Development Life Cycle Models", Journal IEEE Transactions on Software Engineering ,Vol. 14, Issue 10, 1988.
15. Maglyas, A.; Nikula, U.; Smolander, K.,"Comparison of two models of success prediction in software development projects", Software Engineering Conference (CEE-SECR), 2010 6th Central and Eastern European on 13-15 Oct. 2010, pp. 43-49.
16. Gaurav Kumar, Pradeep Kumar Bhatia," Impact of Agile Methodology on Software Development Process"," ISSN 2249-6343International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 2, Issue 4",august 2012.
17. Guy Want, "Drowning in the Waterfall, The Benefits of Agile versus the Predominance of Waterfall "Software Engineering CS 390:", October 29, 2008.