Computing, Information Systems, Development Informatics & Allied Research Journal
Vol. 8 No. 2, June, 2017 - www.cisdijournal.net

Comparative Analysis of Structured Approaches with Agile Approaches for Software Development

Usman, O.Y. & Ogwueleka, F.N.
Computer Science Department
Nigerian Defence Academy
Kaduna, Nigeria
Almizansoftwaresolution1@yahoo.com, ogwuelekafn@gmail.com

ABSTRACT

Software development life cycle is a set of structured activities that is required for the development of high quality and reliable software product. There are quite a number of approaches employed in software development. The choice of any of these approaches varies from organization to organization. This paper focuses on only two of these approaches, namely: structural approach and agile approach. The pros and cons of each approach, the software development life cycle and software engineering process were discussed followed by the comparative analysis of the structured and agile approaches. In this paper, Extreme programming and waterfall models were studied in order to make an analysis of the structured and agile approaches. The studied revealed that the agile approach to software development is ideal when the project is small and the budget is small. The implication of the research and practice is the constant communication, and collaboration between the software developer and the customers in the production of working software.

Keywords: Software development life cycle, software engineering process, agile approach, agile manifesto, structured approach

CISDI Journal Reference Format
Usman, O.Y. & Ogwueleka, F.N. (2017): Comparative Analysis of Structured Approaches With Agile Approaches for Software Development. Computing, Information Systems, Development Informatics & Allied Research Journal. Vol 8 No 2. Pp 85-92
Available online at www.cisdijournal.net

1. INTRODUCTION

Software Development Life Cycle (SDLC) is a set of structured activities (essentially a series of steps, or phases) that provide a model for the development of a high quality and reliable software product (Ashish , 2015; Sanjana and Shaveta, 2011). Software development model is an abstract representation of processes describing the approaches of a variety of activities that take place during the software development process (Sanjana and Shaveta, 2011). The varieties of models are waterfall, extreme programming (XP), spiral, iterative, scrum and prototype. Software Engineering method refers to structured approaches to software development which include system models, notations, rule, design advice and process guidance (Nwaocha, 2009). Software engineering (SE) is the application of systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software because it integrates significant mathematics, computer science and practices whose origins are in Engineering (Vanshika, 2015). There are quite a number of approaches employed in software development. The choice of any of these approaches varies from organization to organization. These approaches include the structured approach, object-oriented approach and a current trend approach known as agile approach. This paper will only focus on the structured and agile approaches.

2. REVIEW OF RELATED STUDIES

Gaurav (2014) carried out a research on the comparative analysis of software engineering models from traditional to modern methodologies. The research discussed the Software Engineering model such as traditional method and the modern method. The work gave examples of each of the methods, advantages, their shortcomings and comparison between the two methodologies. The work concluded that the development team is to select a software development model that best suit the project. Nabil et al. (2010) presented a research on comparison between five models of software engineering, discussing the strength and weakness of the following models namely: waterfall, iteration, V-shaped, spiral and Extreme Programming. The main objective of the research is to present different models of software development phases and make comparison between them to show the features and defects of each model. Vanshika (2015) carried out research on software development life cycle model – comparison and consequences. The research deals with five models namely: waterfall, iteration, V-shaped, spiral and Agile model. The work further made comparison among various SDLC models. Seanna and Sona (2012) provided a framework which serves as a basis for analyzing the similarities and differences among different life cycles model. The research examines the various models and made a comparison between the features and defects of the models. Malik and Siew (2009) in their work reviewed the agile methodologies in software development.

According to their research, difficulties in development of software usually emerged as the companies grow. These difficulties include evolving requirements, customers involvement, miscommunication, tight deadline and inadequate budget. Agile methodologies include Extreme Programming. Agile Modeling, SCRUM, Crystal methodologies family, Feature-Driven Development, and Adaptive Software Development. The work only discussed only three of the aforementioned agile methodologies that widely used in software development. They include Crystal methodologies, Feature-Driven Development, and Adaptive Software Development. The research revealed that the agile methodology have some limitation of documentation, dependence of user involvement, the quality of the product depends solely on the behaviors and skills of the developer. They concluded that when deadlines and budgets are tight, Agile methodologies are one of the best software development approaches to apply. Simran et al. (2015) in their research discussed how technology team, work together to plan, build and deliver software product before the deadline. The manifesto comprises of such as individuals and interactions; working software; customer collaboration and responding to change. The Agile Manifesto is based on twelve (12) principles and the five phases of the agile methodology discussed were feasibility, planning, development, adapting, and deployment. The Agile development provides opportunities to assess the direction throughout the development lifecycle through iterations.

Kuda et al. (2011) in their work discussed the prevailing agile methods and practices, applicability, pros and cons of agile methods. According to them, agile methods focus on customer satisfaction by structuring the development process into iterations where in each iteration produces sizeable amount of working code and artifacts of interest to customers. Also they asserted the because of fast changing technology, the customers found it difficult to define their needs. New methods, now called agile approaches are were designed to define the changing requirements in software development environments. Agile manifesto was also discussed. Examples of agile methodology such as crystal methodologies, Dynamic Software Development Method (DSDM), Lean software, Scrum and extreme Programming were also given. In their findings, the reasons motivating to adopt agile methodology are ability to adapt to change, ability to get instant feedback from customer, pressures demand short time frames and organizational processes demand high quality bug free software. Rupali (2015) research focused on traditional method such as the Waterfall model and comparative study of agile method. A typical software development life cycle consists of the following stages: According to him, the popular SDLC models followed in the industry including Waterfall Model, Iterative Model, Spiral Model, V-Model, Big Bang Model. He discussed the twelve principles of the Agile methodology.

3. STRUCTURED APPROACHES

Structured approach is the method which is used in the analysis of information for system development . The three types of techniques that are used in the structured approach are the structured analysis, structured design and the structured programming. A structured technique can result in structured diagrams such as control logic flowcharts, hierarchy diagrams, business process flowcharts, organizational flowcharts, data-flow and entity-relationship diagrams, and other processes flowcharts, to mention a few (Marion, 2001).

The structured systems analysis and design methodology (SSADM) has been developed and mainly used by government departments. This is why it is mainly designed for large-scale Information Systems with high volume business events. SSADM specifies exactly the flows and tasks of a development project and produces a detailed documentation of the project. SSADM sticks to the traditional waterfall model, which allows review of each stage but requires its accomplishment before the next one can begin. SSADM focuses on the analysis and design stages of the systems development life cycle (SDLC). SSADM combines three methods, complementing each other within a systems development cycle: Logical Data Modelling, Data Flow Modelling, Entity Event Modelling (Marion, 2001). Structured methodology is based on Waterfall model.

3.1 Development Stages of SSADM

The following are the stages of SSADM as discussed by Marion (2001)

Stage 0: Feasibility study
Investigation of economical and technical feasibility. The problems are defined and the project identified. The best business option is chosen out of up to 5 propositions. In SSADM the feasibility stage is not imperative.

Stage 1: Investigation of the current environment
Definition of broad requirements, investigation of current data and processing. The project is being identified and costs calculated. This stage is especially important as any omissions will have a bad effect on the whole project.

Stage 2: Business Systems Options
Formulation of business systems requirements. Evaluation of the implication and benefit of each proposed option.

Stage 3: Requirements specification
Identification of functional and non-functional require- ments in detail. Proposal of new methods and techniques in order to describe processing and data structures. Produce a logical design of the proposed systems.

Stage 4: Technical systems options: Definition and selection Maintenance of specific technical options, such as different methods of implementation. and Review

Stage 5: Logical design: May be simultaneously to stage 4. User dialogues, update processes, enquiry processes are defined and selected.

Stage 6: Physical design: After producing a physical design, creating a function and data design, the SSADM cycle is completed and the applications are ready for delivery.

3.2 Benefits of SSADM

According to Marion (2001) the benefits of SSADM include It allows the product to be planned, managed and control in order to deliver the product on time. Other benefits of SSADM include usability, respond to changes in the business environment, effective use of skills, better quality, improves productivity, and cost reduction.

3.3 Weaknesses of the Structured Approach

The weaknesses of the structured approach as discussed by Marion (2001) include it aaddresses some but not all of the activities of analysis and design. Structure approach makes processes rather than data the central focus.

3.4 Waterfall Model

Waterfall Model is referred to as a linear-sequential life cycle model. Waterfall Model is simple to understand and flexible use. Testing can only be performed only when the software is fully developed. In a waterfall model, there is no overlapping in phase each phase must be completed before the next phase can begin (Rupali, 2015). The waterfall model is shown in Figure 1.

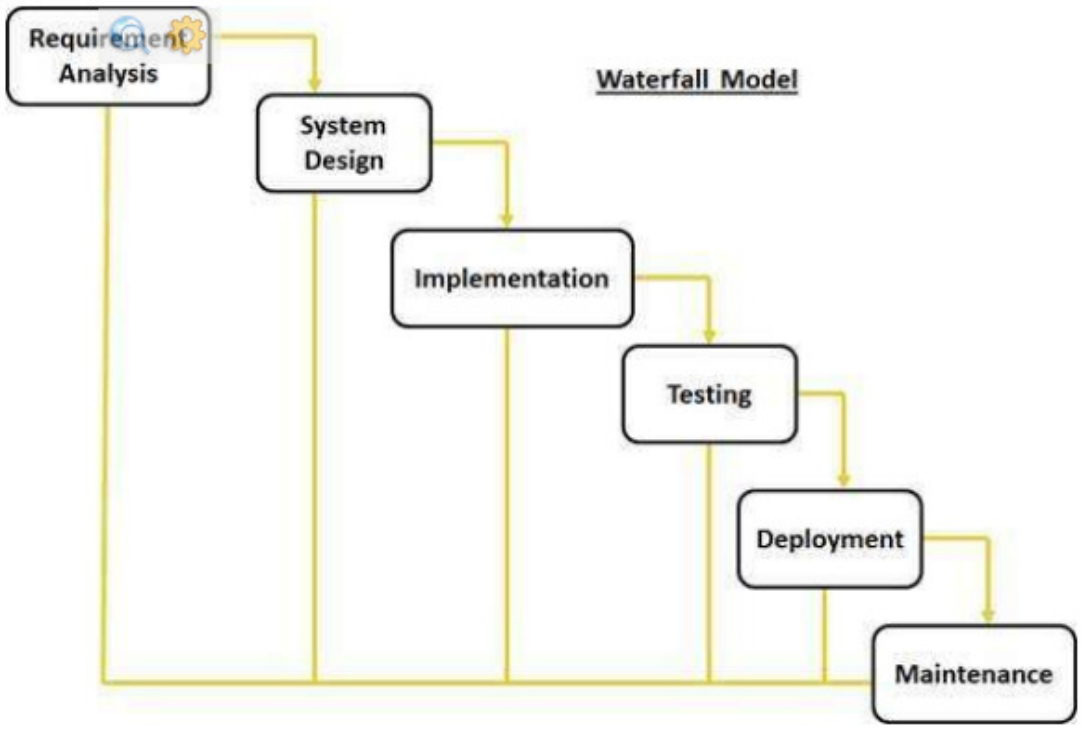

Figure 1: Graphical illustration of the waterfall Model, (Source: Rupali, 2015)

3.5 Pros of Waterfall Model

According to Rupali (2015) before the development commences requirement is clear. Each phase of the development is completed within a specified period of time after which it moves to next phase. It is easy to implement. The amount of resources needed to implement this model is minimal. Documentation of each phase is done for the quality of the development. Waterfall model is simple and easy to understand and use, easy to manage due to its rigidity (Sahil et al, 2015).

3.6 Cons of Waterfall Model

According to Rupali (2015) problems with one phase cannot be fixed completely during that phase. When a client wants a change to requirement to be made, it will not be implemented in the current development process. Shubhmeet (2015) asserted that there is no user involvement, life cycle is too long. Waterfall model has high risk and uncertainty, It is difficult to measure performance within phases, not ideal for complex and object-oriented projects, no working software is produced until late in the life cycle (Sahil et al., 2015).

3.7 Area of Application

The waterfall model is used when
1. Requirements are fixed and known
2. Project is of short duration.
3. Automating any existing manual system. (Shubhmeet, 2015)

4. AGILE APPROACHES

Agile development is a philosophy in which a set of guidelines for developing information systems in an unknown, rapidly changing environment, and it can be used with any system development methodology (Approaches to System Development system, 2011). Agile modeling is an idea about how to build models, some of which are formal and detail, others sketchy and minimal (Approaches to System Development system, 2011). Agile model is referred to as several iterative and incremental software developments. Examples of this model include Extreme Programming (XP), Scrum, Crystal Dynamic System etc. each of these have a specific approach, but common vision and core values (Ian, 2011). Agile software development is a style of software development that emphasizes customer satisfaction through continuous delivery of functional software" (Vanshika, 2015). Agile methodology is an iterative and incremental process (Simran et al., 2015).

According to Erickson et al. (2009) agility means to strip away as much of the heaviness, commonly associated with traditional software development methodologies, in order to promote quick response to changing environments, changes in user requirements, accelerate project deadlines, and the like. Agile development approach gives the developer opportunity to assess the direction development life cycle through regular Sprints or iterations, at the end of which a functional software increment could be produced. Agile Manifesto was created in 2001 by a number of IT professionals in a conference held in Utah to develop new approaches to software development. They developed the new approaches to software development base on the rule that the best approach to verify software is to deliver working versions of the software to the client i.e. the customer, and then update it based on their observations and notes (Malik and Siew, 2009). The agile approach is based on four principles such as to develop a system that meets the customers' requirement, accept changes in requirements at any stage of the system development, synergy between the developers and the customers throughout the period of the system development, and developing the system on a test-driven basis (Malik and Siew, 2009).

4.1 Types of Agile Methodology

There are several types of agile methodology. Some examples include Extreme Programming. agile Modeling, SCRUM, Crystal methodologies family, Feature-Driven Development, Adaptive Software Development. Out of these examples only the extreme programming will be discussed for the purpose of this work.

4.1.1 Extreme Programming (XP)

XP is a type of agile methodology that produces high quality software in a short development cycle and introduce checkpoints where new customer requirement can be adopted (Rupali, 2015). XP accepts change at any time of the development life cycle. The main features of extreme programming are small releases, continuous feedback, refactoring is performed at end of each stage rather than at the end of the process to correct bugs, pair programming, customer is available at all times to address issues and assign priorities, integration and testing is performed multiple times (Shubhmeet, 2015).

4.1.2 Pros of XP

It could be used for small project (Kuda et al., 2011). It accepts change at any stage of the development. According to Shubhmeet (2015) the advantages of agile model include quick releases, meets changing requirements, emphasis is on customer feedback, measures real-time performance, less overhead, bugs are identified and removed immediately, changes are less costly, there is improved coordination among programmers during pair programming, close customer collaboration.

4.1.3 Cons of XP

It is not suitable for one-developer-projects. Customer participation is minimal. Testing of the development is performed by the same person as such possible bugs may not be detected since the developer tests from the same perception the product is developed (Kuda et al., 2011).

4.1.4 Area of usage of Extreme Programming

Extreme programming is use where requirements changes frequently, releases must be quick, development and changes is allowed anytime without documentation (Shubhmeet, 2015). The agile model is shown in Figure 2

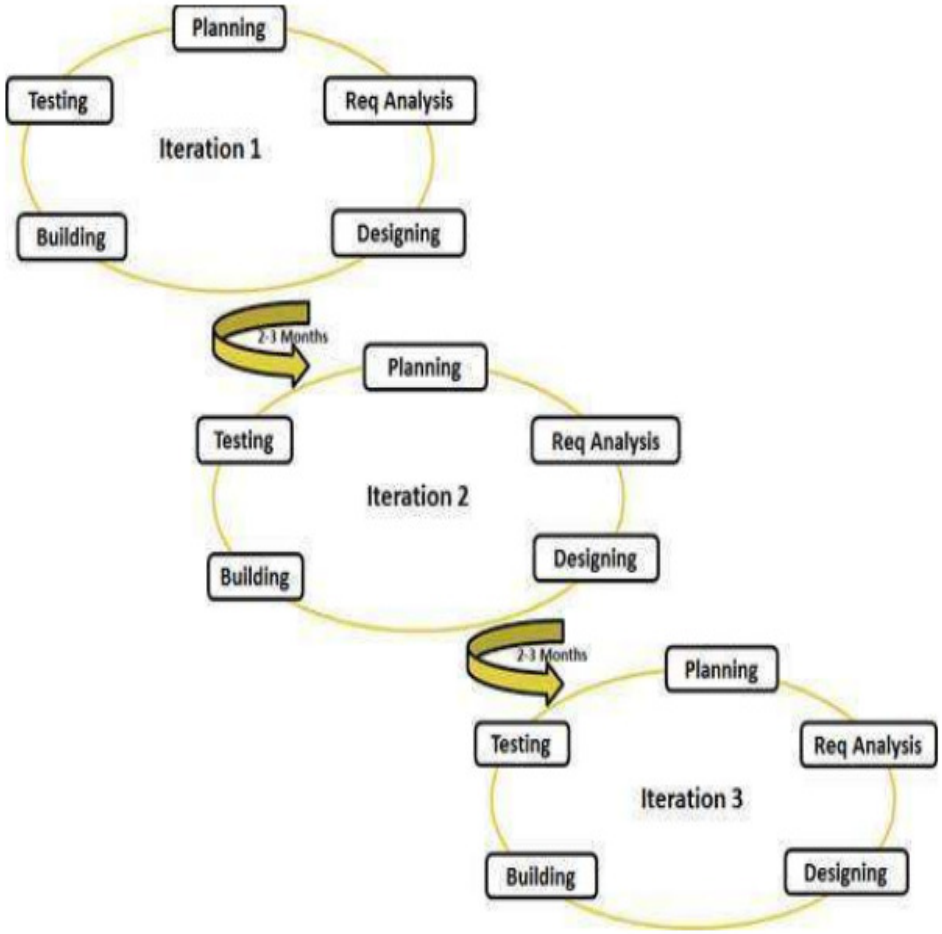

Figure 2: Graphical illustration of the Agile Model, (Source: Rupali, 2015)

4.2 The Agile Manifesto

In February 2001, 17 software developers met at the Snowbird resort in Utah where they discussed some lightweight software development methods. At the end of the conference, they published the Manifesto for Agile Software Development. The manifesto comprises of some concepts which are described as below as discussed by Simran et al. (2015):

i. Individuals and interactions: self-organization and motivation are important, as are interactions like co-location and pair programming.
ii. Working software: working software is more useful and welcome than just presenting documents to clients in meetings.
iii. Customer collaboration: requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important.
iv. Responding to change: agile methods are focused on quick responses to change and continuous development.

4.3 Feature of Agile Approach

the features of agile approach as discussed by Ian (2011) are that it supports incremental development, supports frequent system releases, encourages customer involvement with the development team, supports change through regular releases and maintains simplicity through constant refactoring of codes.

4.4 Process of Agile Software Development

The Process of Agile Software Development involves the following (Vanshika, 2015):

1. Starts with a kick-off meeting.
2. The known requirements are understood and prioritized. The development is plan is drawn accordingly.
3. Relative complexity of each requirement is estimated.
4. Sufficient design using simple diagrams is done.
5. Test Driven Development (TDD) approach may be used. TDD emphases on "Writing Test First and then Writing Code to Pass the Test". It can help in avoiding over-coding.
6. Development is done, sometimes in pairs, with lot of team interaction. Ownership of code is shared when pair programming is done.
7. The code is tested more frequently. Sometime a dedicated "Continuous Integration" Server/Software may be used to ease the integration testing of the code.
8. Depending on the feedback received, the code is refractor. Refactoring does not impact the external behavior of the application but the internal structure may be changed to provide better design, maintainability. Some ways of refactoring may be add interface, use super class, move the class etc.

4.5 Advantages of Agile Methodology

The pros of the agile methodology are that it **e**mphasizes final product, iterative, produces good team cohesion, lightweight methods suitable for small size projects, test based approach to requirements and quality assurance.

4.6 Disadvantages of Agile Methodology

Where documentation of the development process is necessary, it is difficult to scale up to large project. It requires an experience and skill programmer, programming pairs is high (Nabil, 2010)

4.7 Agile Modeling Principles

According to Rupali (2015) the agile methodology is based on the following twelve (12) principles:

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

5. COMPARISON BETWEEN STRUCTURED APPROACHES AND AGILE APPROACHES

Table 1 is the comparison between Structured Approaches and Agile Approaches

Table 1: Comparison between structured approaches and the agile approaches.

Features	Structured Approach	Agile Approach
Requirement specification	Beginning	Defined incrementally
Cost	High	Low
Risk analysis	High	Low
Expertise requirement	High	High
Guarantee of success	High	High
Implementation time	Long	Short
Flexibility	Rigid	Flexible
Maintenance	Least	Easy
Users involvement	Low	High
Simplicity	Simple	Simple
Changes incorporated	Difficult	Easy
Availability of reusable component	Limited	High
Overlapping phase	No	No
Understanding requirement	High	Low
Development direction	fixed	Flexible
Integrity and security	Least	Robust
Documentation and training requirement	Vital	Yes
Customer satisfaction	Low	High
Performance	Low	High
Project type	Large-scale	Low to medium scale
Fault detection	Problematic	Easy
Testing	After coding	On every iteration

6. RESULT OF FINDINGS

From the study, we understood that many developers adopt the agile methodologies. Agile methodologies adapts quickly to the changing requirements trend of the customers. It maintains a high quality standard throughout the design and testing of the software. As software is released in intervals, bugs can easily be detected and fixed. Other the other hand the structured approach is not flexible to change. It can only be used for large projects. The structural approach can be used only when requirement are defined.

7. IMPLICATIONS FOR RESEARCH AND PRACTICE

From our research several implications for research are evident:

- Working software: the highest priority is to satisfy the customer through early and continuous delivery of working software.
- Customer collaboration: Requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important. there should be synergy between the software development team and the customer so that a working software free of bugs will be delivered..
- Communication: the developer and the customer need to meet and discuss the overall objectives of the software.

8. CONCLUSION

Software development life cycle (SDLC) is a structured set of activities required for the development of a high quality and reliable software product to meet a business need. After completing the research, it is concluded that the agile approach would be appropriate if the project is small and required with a small time frame while the agile approach would be ideal if the budget is small. The structured approach is usually adopted by large organizations such as government, and the structured approach is not only bureaucratic but time and cost consuming.

REFERENCES

1. Ashish, K. G. (2015). "A Comparison Between Different Types of Software Development Life Cycle Models in Software Engineering", *International Journal of Advanced Technology in Engineering and Science*, 3 (1); 624 – 631.
2. Gaurav, K. (2014), "Comparative Analysis of Software Engineering Models from Traditional to Modern Methodologies". *Fourth International Conference on Advanced Computing and Communication Technologies* DOI: 10.1109/ACCT 2014.73
3. Ian, S. (2011). "Software Engineering", Addison Wesley, 9th edition; 18 – 100
4. Kuda N. R, Naidu, G. K., and Praneeth, C. (2011) "A Study of the Agile Software Development Methods, Applicability and Implications in Industry". *International Journal of Software Engineering and Its Applications* 5 (2); 35 – 45
5. Malik, H. and Siew, H. O., (2009) "Review of Agile Methodologies in Software Development", *International Journal of Research and Reviews in Applied Sciences* 1(1); 1 – 8
6. Nabil, M., Ali, M. and Govardhan, A. (2010) "A Comparison between five Models of Software Engineering". *Internal Journal of Computer Science Issues*, 7 (5); 97 – 101.
7. Nwaocha, V. (2009), "Software Engineering Methodologies" National Open University of Nigeria, Lagos, 1st Edition; 8 – 24
8. Rupali, P. P. (2015), "A Comparative study of Agile Software Development Methodology and traditional waterfall model " *IOSR Journal of Computer Engineering* 1 (1); 1 – 8.
9. Sahil, J., Puneet, G., and Praveen R. (2015)," Various Software Development Life Cycle Models", *IJRDO - Journal of Computer Science and Engineering*, 1 (4); 162 – 167.
10. Sanjana, T. and Shaveta, G. (2011), "Comparative Analysis of Software Development Life Cycle Models", *International Journal of Computer Science and Technology*, 2 (4); 536 – 539
11. Seanna, T., and Sona, M. (2012), "Analysis and tabular comparison of popular SDLC Models" *Internal Journal of Advances in Computing and Information Technology*, 1 (3); 277 – 286.
12. Shubhmeet, K. (2015), "A Review of Software Development Life Cycle Models", *International Journal of Advanced Research in Computer Science and Software Engineering*, 5 (11); 354 – 360.
13. Simran, B., Vandana, T. and Pooja G. (2015), "Agile Software Development", *IJRDO - Journal of Computer Science and Engineering* 1 (4); 108 – 112.
14. Marion Schumacher (2001) "The use of SSADM (Structured Systems Analysis and Design Methodology) as a standard methodology on Information Systems Projects"; 1 – 10 available at http://www.hausarbeiten.de/faecher/vorschau/106034.html accessed on 6 June, 2017
15. Vanshika, R. (2015) "Software Development Life Cycle Models-Comparison, Consequences". *International Journal of Computer Science and Information Technologies,* 6 (1); 168 – 172.