BIU Journal of Basic and Applied Sciences 3(1): 80 – 91, 2017.

ISSN: 2563-6424

A COMPARISON BETWEEN FOUR SOFTWARE PROCESS MODELS

***IYAWE, G. O., AKHIDENO, I. A AND IGBINIGIE, P-C.**
Department of Physical Sciences, Benson Idahosa University, Benin City, Nigeria,
*Corresponding author: gitegboje@biu.edu.ng

ABSTRACT
Over the years, developers have encountered some challenges with software development. Software researchers have been able to develop process models that cater to the different types of software application and if followed correctly can result in software that are delivered on-time, cost effective and conform to user's needs. Knowing the right process model to use in software development can posed a challenge for software developers. This research deals with Software Development Process and briefly describes four process models which are: Waterfall Model, Spiral Model, Rapid Application Development (RAD) and Extreme Programming (XP), outlining their advantages and disadvantages and making a comparison to show the features and flaw of each model. These models were chosen because they represent the Tradition lifecycle models and the Agile process model.

KEYWORDS: Software development, Software Engineering, Software Lifecycle, Process Models

INTRODUCTION
The development of quality software with little or no faults has been a great challenge to the computer society, and the NATO's 1968 conference held in Germany was as a result of a rapid increase in the size, complexity of software systems and the inability of the software industry to provide the user with high quality software on time. As software projects and the size of systems grew bigger in scope, more than one person was required to develop the system. Therefore, project teams were needed to carry out large software development. Managing the development of these large systems also became a problem, it was observed that most of the software did not meet their requirements, some were delivered late, some exceeded their budgets and others were cancelled. There was a need to ensure that these problems were eliminated, hence the introduction of a more disciplined and structured approach to software development. It was during the conference that the discipline "Software Engineering" was invented to study the process of developing large and reliable software.

According to IEEE, Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation

and maintenance of software, that is, the application of engineering to software. Sommerville (2004). From this definition, it can be noted that software development is a step by step process and in a predetermined order, and to ensure that the development process can be measured, a process model is needed so that software products are developed in the right order. When using a process model; people, tools, methods, requirements and standards are used to develop the product.

Software Development Process

This section discusses the Software development process which includes the four fundamental activities in software development. Each Phase produces deliverables required by next phase in the Software development.

Software systems are often large and complex, therefore, there is a need to be organised when carrying out a development process. For long, the process of developing large software systems has been a huge challenge. The Standish Group Report (1995) identified some of these challenges as incomplete set of software requirements, lack of user involvement, insufficient budget and schedule overruns.

Software process is a framework for the tasks that are required to build high quality software Pressman (2009). These tasks may include the definition of requirements, design of software products, programming, implementation, operation, maintenance and evolution. It is apparent that software goes through a series of processes starting from their inception, initial development, operation, maintenance until they eventually retire. There is no ideal software process for software development. This is due to factors such as differences in software system as a result of the goals of the system, nature of the development team working on the software system, amongst other factors. However, software development process permits technologist to work together using different approaches to develop quality software. A software process is chosen base on the nature of the system being developed for example, safety critical systems or systems with stable requirements would require a very structured software process (plan-driven) while systems experiencing constant changes to requirements for example, business systems would require a less formal (Agile) software process.

According to Booch and Rumbaugh (1999) “A software process defines who is doing what, when and how to reach a certain goal”. There are several software processes and but there are the four fundamental software activities that are common to these process Somerville (2004). These activities are:

a. Specification
b. Development
c. Validation
d. Evolution

Specification

Specification is the first phase of software development. In this activity, the customers and developers define the software that is to be produced and the constraints on its operation.

The specification phase comprises of the following:

i. Problem Definition
ii. Feasibility study
iii. Requirement gathering/Analysis

During the problem definition phase, the problem is stated and defined by the

user to the software project manager, a feasibility study is then carried out so as to gather the requirements for the proposed system, which is further analysed by a requirements analyst or the development team.

Software Development

In this phase, the software is designed and constructed. System modelling and the software programming also take place here. The requirements are used in the software development. The development process involves design and coding the system. The software designers are responsible for ensuring the interface required for the system are designed using the appropriate design tools, eg. Flowchart, UML, and CASE tools, a prototype of the actual system can also be designed. Implementation is done to translate this structure into executable program.

Validation

This is where the software is checked to ensure that the goals for which the systems were developed have been reached. In this activity, Verification and Validation are carried out to on the software system by both the development team and the client to ensure that it meets the users' needs and conforms to the requirements.

Software Evolution

This is the last phase of the software development activities, this phase encompasses maintenance and evolution**.** Software Evolution is the dynamic behavior of programming systems as they are maintained and enhanced over their lifetimes. The development of software continues even after its deployment. Large systems with potentials of having longer life spans need to experience changes in their lifetime to remain useful. The changes that can affect any software may be as a result of the following factors.

i. Changes in requirements to reflect a changing business, technology, stakeholder or user needs.
ii. Changes in the environment of the system due to the introduction of a new hardware.
iii. System repair to remove/correct errors in coding or design. This could also be referred to as system maintenance.
iv. A need to add new capabilities that were previously thought to be impossible.

Description of Software Process Models

Sommerville (2010) define Software Process Model as an abstract representation of a process. It's a step by step process that ensures that the various activities of Software development are carried out in the right order.

Software Process Models can either be Plan Driven or Agile Process. Plan Driven Processes are usually heavily documented and follow a plan e.g. Waterfall Model while Agile Processes are Flexible and can adapt to changes e.g. Extreme Programming (XP).

There are several software process models which are:

i. Linear Sequential Process Models: Waterfall model- Specification and Development are usually separate and distinct.
ii. Evolutionary/Iterative Models: e.g. Spiral Model, Prototyping Models, RAD, and Concurrent Development Model. – Specification and Development are usually interleaved. Specification. Initial

system rapidly developed and then refined with customer input.

iii. Incremental Model: Specification, development and Validation are interleaved. It may be Plan Driven or Agile.

iv. Component Based Models: These are reusable components which are integrated into a system. i.e., they are assembled from existing components. They may be Plan driven or agile.

Comparison of four process models

This section focuses on four Software Development Process Models, outlining the advantages and disadvantages of each model and the suitable cases in which these models can be applied. These models are;

i. Waterfall Model
ii. Spiral Model
iii. Rapid Application Development (RAD)
iv. Extreme Programming

These models were chosen because their features correspond to both plan driven and agile processes.

Waterfall Model

Lifecycle model development began in 70's with suggested Waterfall approach as a pioneer in software development. It is a disciplined, structured and well-defined model that is widely used in software engineering. This model was proposed by Winston W. Royce and was successfully used in the industry during these years. The first known presentation describing use of similar model phases was held by Herbert D. Benington at a Symposium on advanced programming methods for digital computers in 1956. Royce (1970) first described the "Waterfall model" in an article as a model in which additional steps need to be added for it to be an effective working model. However, he never used the term "Waterfall Model" throughout the article. The waterfall model mirrors the linear sequential (traditional lifecycle) methodology and is typically used in highly structured environments in which safety and security is the number one requirements for a system and also for systems that are complex, mission-critical and future changes are costly, if not impossible. The Specification and Development phases are usually separate and distinct.

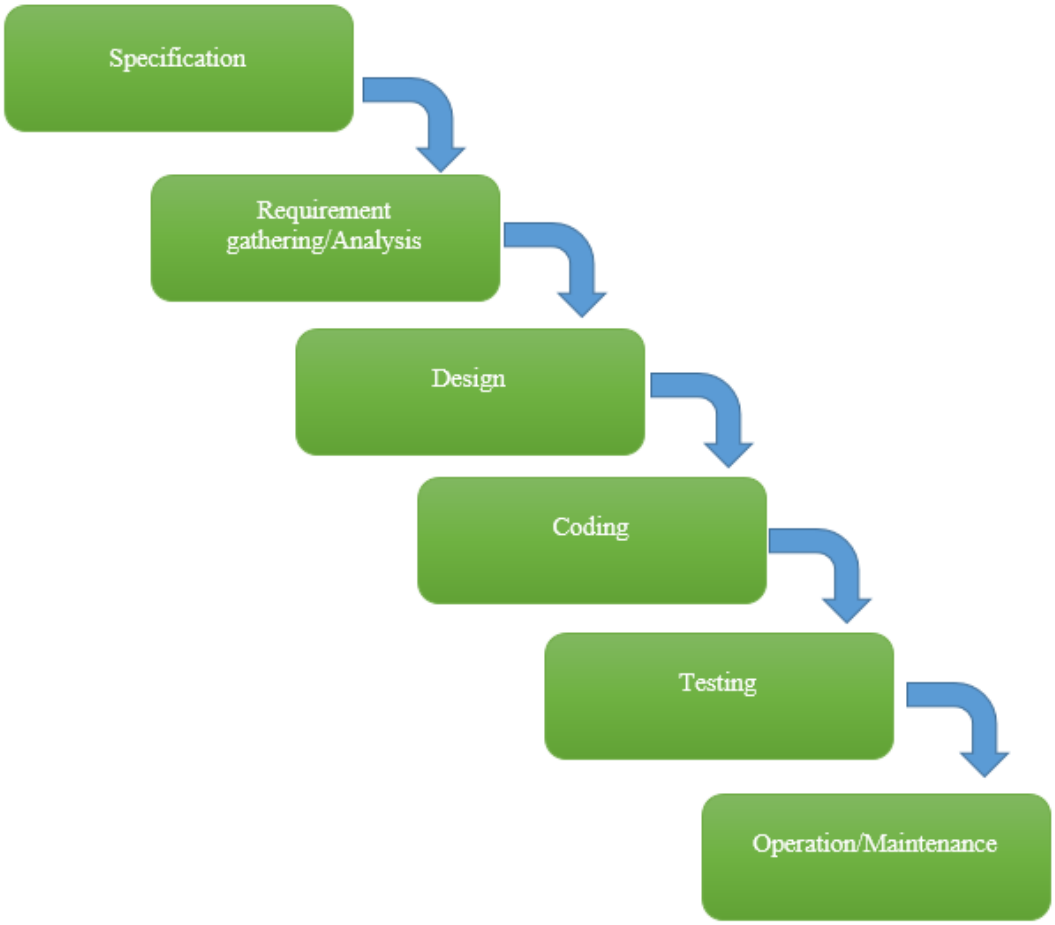

Fig 1: Diagram of the Waterfall Model

The Waterfall Model is often represented as a step-by-step linear structure as shown in fig 1. It involves sequential task to be completed and well scripted milestones. Each step has to be completed before moving to the next step and the entire functionality is developed and then tested altogether at the end. The idea behind this model is to gather requirements and analysis at the early stages to save time and effort later and continue development, which is planned out from the beginning until completion. In Waterfall model, each phase is extensively reviewed and documented so as to prevent returning to a previous phase as this can be costly.

Advantages

i. It reinforces good standard, i.e. ensuring that a phase is thoroughly completed and must meet good standard before proceeding to another phase because the deliverables of one phase is the input of the next phase.
ii. It ensures that milestones and deliverable are identified and met.
iii. It is easy to understand and implement
iv. It results in certainty at each phase.
v. Waterfall model is documentation driven. It lays emphasis on

documentation such as requirement and design documentation including the source code. In waterfall model, every phase calls for documentation to be done, this helps the development team in the next phase. Documentation is very important because it provides information and a clear description. It helps track the flow of a system.

Disadvantages

i. It freezes requirement which makes it difficult to accommodate changes
ii. It is rigid and cannot accommodate inevitable changes
iii. It can only be used for large projects as it is costly for small projects.
iv. Software is usually delivered late.

Spiral Model

The failure of the Waterfall model to address most of the issues that were present in software development, which was mainly evolving, and changing requirements led to the spiral model, which included improvement of the former model and aspired to overcome the limitations of the waterfall model.

In 1987, the defence science Board Task Force Report on Military software emphasized the concern that the traditional process models were discouraging effective approach to software development such as prototyping and software reuse.

Barry Boehm proposed the Spiral Model, it is a risk-driven model that incorporates the strengths of other models such as the waterfall model and resolves their difficulties. It creates a risk-driven approach to the software process rather than a primarily document-driven process.

The spiral model was designed to include the best features from the waterfall model and the prototyping models and also to introduce a new component-risk-assessment. Risk assessment is included as a step in the development process to evaluate each version of the system. The project can be aborted if the customer decides that the identified risks are too great.

In the spiral model, each version is developed based on the steps of the waterfall model. An initial version of the system is created and presented to the customer. Based on the customer evaluation and feedback, repetitively modified versions are developed. The process of iteration continues throughout the life of the software until the customer is satisfied. The spiral model address risk reduction, it plans ahead of the risks.

The first thing to do before embarking on a new loop is to decide what the major difficulties to be handled are. Spiral model provides a rapid development and at the same time incremental versions of the software application. It is iterative, and each repetition is designed to reduce risk.

The Microsoft operating system is an example of the application of the spiral model. There was an evaluation of the Microsoft windows operating system from Windows 3.1 to Windows 2003. The Microsoft Window 3.1 operating system was the first iteration; the product was released and evaluated by the customers. After getting the feedback from the customers, Microsoft developed a new version of windows operating system. Windows' 95 was released with the enhancement and graphical flexibility and since then other versions of windows operating system has been released.

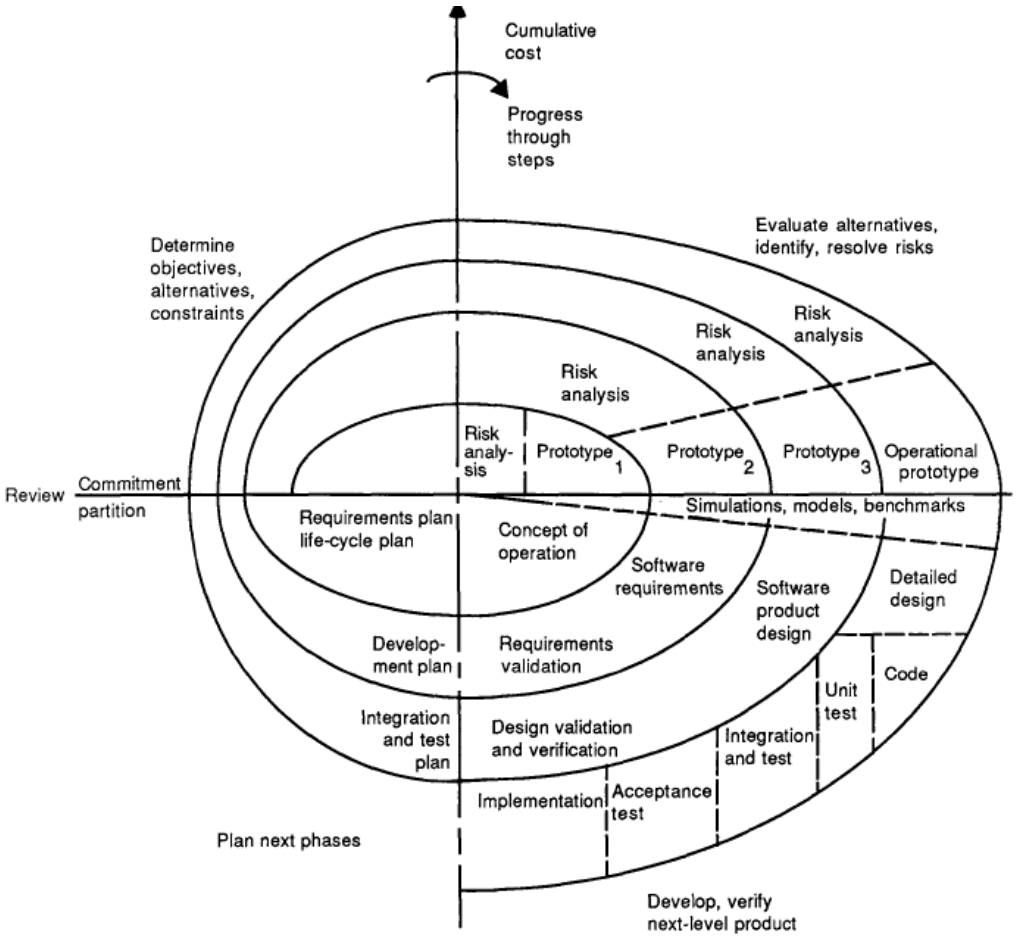

Fig. 2: Spiral Model
Source: Boehm (1988)

The Spiral model consists of four phases as shown in figure 2 above, which are: Planning, Evaluation, Risk Analysis, and Engineering. Each loop in the spiral represents a phase of the software process e.g. system feasibility, requirements definition, etc. Each loop split into four sections: Objective setting, Risk assessment and reduction, Development and validation and Planning.

Each cycle in the model begins with the identification of the objectives of the product to be developed, certain factors need to be considered, e.g. Functionalities and performance, then an alternative(s) means of implementing this product is determined and the constraints imposed on the application of these alternatives are identified. The alternatives can include, the cost of developing the product, schedule etc. Areas of uncertainty that can cause risks are identify and resolved using cost effective strategies or risk resolution techniques such as Prototyping, questionnaires etc. Once these risks have been resolved, a prototype is developed, the software is produced and validated and the project continues to the next spiral.

Advantages

i. It is risk driven
ii. It can used for large and safety-critical systems
iii. Software is developed early in the software lifecycle.
iv. It is flexible and allows changes to be implemented at different stages in the development.

Disadvantages

i. It is time consuming and expensive.
ii. It is not suitable for small teams
iii. Risk analysis requires team members with high expertise.

Rapid Application Development

Rapid Application Development was developed by James Martins in 1991. It is an incremental Software development process model and one of the most widely used and successful implementation of prototyping. It is suitable for information systems, web-based, and e-commerce system and aims to shorten the lifecycle and produce system quickly to accommodate increasing changes in requirements.

Martins (1991) defines RAD as a generalised lifecycle of a software application development, designed to construct applications much faster, of higher quality than the traditional life cycle.

The RAD life cycle consists of: Requirements planning, user design, construction and cutover. The user is greatly involved in the requirements planning phase, the developers use these requirements for the design phase, and the design phase is based on a prototyping cycle and requires experienced users and developers. The code is generated in the Construction phase and thereafter, testing and user training in the cutover phase. The development are time boxed, and series of functional prototypes are delivered within this time box, and an iterative process continues until the end of the time box or the application is functionally complete. This enable the software to be delivered on time and gives the customers deliverables to see and use.

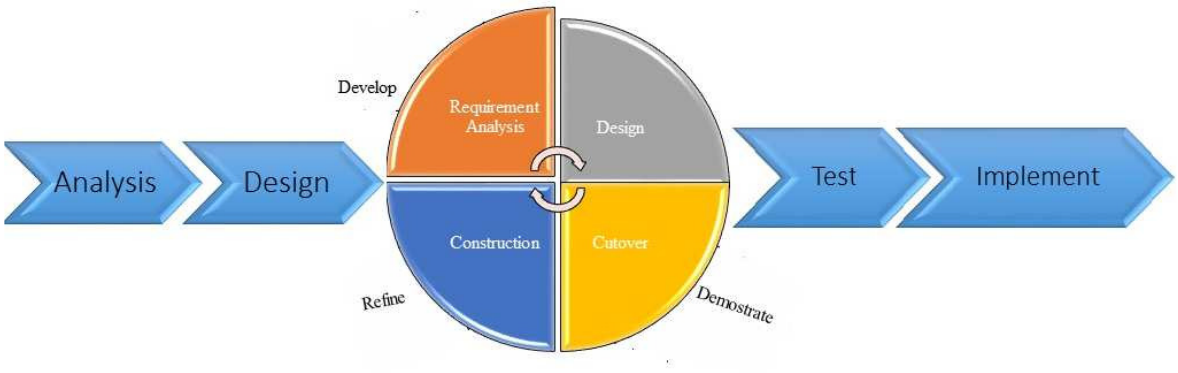

Fig. 3: RAD Process Concept

RAD combines effective tools, techniques, and incremental prototyping. These techniques are: Joint Application Development (JAD), Rapid Prototyping, SWAT team (Skilled with Advanced Tools), Reusability and Time boxing.

Advantages

i. It encourages customer's involvement and feedback.
ii. Time boxing ensures that immediate needs are met and products are delivered on time.
iii. Reduced development time.
iv. It is flexible and can readily adapt to changes in requirements.

Disadvantages

i. It can only work for certain systems e.g. Business systems.
ii. It is dependent on modelling skills.
iii. It requires highly skilled designers and developers.

Extreme Programming

There have been issues with software development especially with existing process models and there is a necessity to produce software that meets user's requirements, delivered on time and within budget. Traditional process models failed to address these issues; they are good at addressing the known and not the unknown. Adapting to changes is one of the many limitations of the traditional process models. The spiral Model meeting budgetary and scheduling requirements is tough given that the iterative development usually spans a long time and is sometimes never ending. Hence the need for process models that are can address the weaknesses of the traditional process models.

The reason behind the development of the agile process models is that, in software development, requirements cannot be fully predicted from the beginning but will always change as the project goes on.

The Agile Manifesto states that:

i. Individuals and interactions over processes and tools
ii. Working software over comprehensive documentation
iii. Customer collaboration over contract negotiation
iv. Responding to change over following a plan

The Agile Process models are: Extreme Programming, Scrum, Feature Driven Development, Crystal, Adaptive Software Development, etc.

Extreme programming was adopted because other existing practices failed to address the issue of changes in requirements. The reason behind the development of XP is that, in software development, requirements cannot be fully predicted from the beginning but will always change as projects moved on.

Extreme programming is a lightweight (or agile) software methodology that is credited to Kent Beck, Ron Jeffries, and Ward Cunningham. Its core values are simplicity, courage, respect, feedback and communication, it is about social changes and addresses most of the limitations of the traditional methods. The technological/business rationale that led to extreme programming was a failed payroll project by Chrysler to determine the best use of object-oriented technologies (Beck 2000).

XP favours collaborations between the development team and the users and ensures that there is continuous feedback from the Stakeholders. Feedback entails: feedback from the system (Unit or Integration testing), feedback from the

customers (Acceptance test written by the users and customers) and feedback from the development team. Fig 4, show the Extreme programming practices.

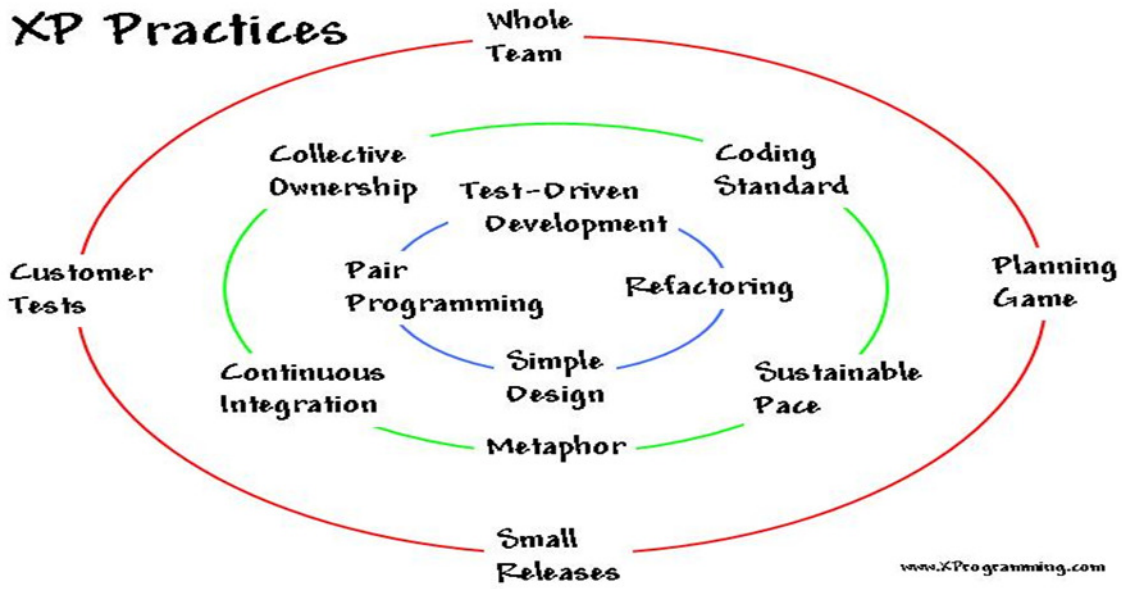

Fig. 4: Extreme Programming Practice
Source: www.xprogramming.com

Advantages

i. Short releases allow the system to evolve.
ii. Having an onsite customer can help in requirement elicitation.
iii. Pair programming can enhance the software quality.
iv. It is cost effective.
v. It is flexible and responsive to change

Disadvantages

i. It is not well defined and does not involve proper documentation.
ii. It may not measure code quality assurance.
iii.

Table 1 Comparison between the four software process models

Model/Features	Waterfall	Spiral	RAD	Extreme Programming
Document/Plan Driven	Yes	Yes	Low	Little/No
Risk Driven	Only at the beginning	High	Low	No
User Involvement	During the requirement phase	High	Yes	Very High (Throughout the development)
Flexibility	Rigid	Low	Yes	Very flexible
Adaptable to changes	Rigid	Yes	Yes	Yes
Cost	Low	Highly Expensive	Low	Low
Overlapping Phases	No (Separate and distinct phases)	Yes (Interleaved)	Yes (Interleaved)	Yes
Scalability	Suitable for large Projects	Suitable for large Projects	Suitable for medium and small scale Projects	Suitable for medium and small scale Projects
Types of Projects	Safety-Critical system	For systems with high risk	Information and web-based systems	Business and information systems
When to use	When Standards are known and Domain will not change	When the projects is high risk	When the system has unstable requirements	When the system has unstable requirements

CONCLUSION

Whether it is the Linear Sequential Process model for the Waterfall methodology or the Components Based Process Model for Extreme Programming and the evolutionary/iterative process model for the Spiral and Rapid Application development models, the choice of process model to be used is usually determined by the nature and scope of the project. Factors such as the size of the project, complexity of the project, budget and requirements all play a key role in the success of the process model chosen and the software quality produced. For safety-critical and large systems, plan driven approach should be used and for medium to small scale systems with changing requirements, agile methods should be used.

REFERENCES

Beck, K. and Andres, C. (2000). *Extreme Programming Explained: Embrace Change*. 2nd ed. USA: Addison-Wesley Professional

Boehm, B. W. (1988). *A Spiral Model of Software Development and Enhancement*, IEEE 61-72.

Boehm, B. W. (2007). *Software Engineering: Lifetime Contributions to Software Development, Management and Research* 2nd ed. USA: John Wiley.

Booch, G., Rumbaugh, J. and Jacobson, I. (1999). The Unified Modelling Language User Guide. Addison-Wesley Professional.

Brookset F. P. (1987). Defence Science Board Task Force Report on Military Software, Office of the Under Secretary of Defence for Acquisition, Washington, DC.

Doolan, M. A, (1996) Rapid Application Development. Hatfield, University of Hertfordshire.

Jeffries, R. E. (1999). What is Extreme programming [online] Available at: <http://xprogramming.com/book/whatisxp> [Accessed: 05 July 2018]

Martins, J. (1991). Rapid Application Development. Macmillian. pp. 81-90.

Nabil, M. and Govardhan, A. (2010). A Comparison between Five Models of Software Engineering. *International Journal of Computer Science Issues (JCSI),* 7(5):94 -101.

Pressman, R. (2009). Software Engineering: A Practitioner's Approach. 5th Edition. McGraw-Hill Higher Education.

Ratnmala, R. R. and Haresh, M. R. (2013). Comparative Study of Various Process Model in Software Development. *International Journal of Computer Applications*, 82(18): 0975 – 8887.

Royce, W. W. (1970). *Managing the development of large software systems.* Available at http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf [Accessed: 03 July 2018]

Sommerville, I. (2004). *Software Engineering* (7th ed.). Harlow, Essex, UK: Addison Wesley.

Sommerville, I. (2010). Software engineering; 9th edition, USA, Addison-Wesley.

Subbarayudu, B., Srija, H. D., Amareswar, E., Gangadhar, R. and Kishor, K. (2007). Review and Comparison on Software Process Models. *International Journal of Mechanical Engineering and Technology (IJMET)*, 8(8): 967–980.

The Standish Group Report, (1995). Chaos, Available at: http://www.projectsmart.co.uk/docs/chaos- report.pdf [Accessed: 03 July 2018]

Wells, D. (2009). Extreme Programming: A gentle introduction [online] Available at www.extremeprogramming.org [Accessed 05 July 2018]