Software Development Methodologies: Agile Model Vs V-Model

Stephen O., Oriaku K.A

***Abstract*— The software development method employed in the development of a software system, play a critical role in the overall software development process. Often time, non-professional software developers jump into a software development project without first choosing an appropriate software development model that best suits the project and that could aid the successful completion of such project. This scenario has led to numerous crappy and abandoned software projects. This research work is aimed at exploring and comparing the two prominent and widely used software development models—Agile and V-model; their impacts in the software development world and general recommendations.**

***Index Terms*— Software Development, Agile Model, V-Model, Extreme Programming, Crystal and Scrum Method**

I. INTRODUCTION

Software is an integral of part of the modern computing world and will continue to be. Likewise the software development models. The software development models are the foundation for software engineering. O decades now, numerous software development models have been introduced, of which only few have survived to be used today. There is, however, an emerging philosophy producing new processes known as "Agile Software Development". This new processes focus more on people's interactions and early development of code than on documentation and planning [3].

This paper introduces and discusses Agile Software Processes and V-shaped model.The process of developing and supporting software often requires many distinct tasks to be performed by different people in some related sequences. When software engineers are left to perform tasks based on their own experience, background, and values, they do not necessarily perceive and perform the tasks the same way or in the same order. They sometimes do not even perform the same tasks. This inconsistency causes projects to take longer time with poor end products and, in worse situations, total project failure.Watts Humphrey has written extensively on software processes and process improvement in general and has also introduced the personal software process at the individual level in his book Introduction to the Personal Software Process (1997).

The goal of a software process model is to provide guidance for systematic coordinating and controlling o the tasks that must be performed in order to achieve the end product and the project objectives. For software development process, a process model defines the following: a set of tasks that need to be performed, input to and output from each task, preconditions and postconditions for each task, and the sequence and flow of these tasks. We might ask whether a software development process is necessary if there is only one person developing the software. The answer is that it depends. If the software development process is viewed as only a coordinating and controlling agent, then there is no need since there is only one person. However, if the process is viewed as a prescriptive roadmap for generating various intermediate deliverables in addition to the executable code—for example, a design document, a user guide, test cases—then even a one-person software development project may need a process[1].

These models are majorly plan driven because of their solicitation and documentation of a set of requirements that is as complete as possible. Based on these requirements, one can then formulate a plan of development. Usually, the more complete the requirements, the better the plan. Some examples of plan-driven methods are various waterfall approaches and others such as the Personal Software Process [21] and the Rational Unified Process (RUP). An underlying assumption in plan-driven processes is that the requirements are relatively static. On the other hand, iterative methods, such as spiral model based approaches [12], evolutionary processes described in [19], and recently agile approaches [45] count on change and recognize that the only constant is change. The question is only of the degree and the impact of the change. Beginning in the mid-1990's, practitioners began finding the rate of change in software requirements increasing well beyond the capabilities of classical development methodologies [10]. The software industry, software technology, and customers expectations were moving very quickly and the customers were becoming increasingly less able to fully state their needs up front. As a result, agile methodologies and practices emerged as an explicit attempt to more formally embrace higher rates of requirements change.

II. CASE STUDY

Our Scenario is to discuss for the requirement given by the client which software development model/method to be employed in developing his project. Let us have a comparative study between Agile Model & V-Model, which model will be effective in the below project and the Pros & Cons of choosing the model.

The client Requirement is to develop a web based application, online booking of ticket for Hotels, Train, bus and flight in a single site. In the site, users can be able to book for hotels, train, bus and flight tickets in advance for up to six months. The existing company's website does not allow for

Manuscript received November 07, 2014.

Okeke Stephen, Department of Computer Science, College of Physical and Applied Sciences, Michael Okpara University of Agriculture, Nigeria., +2348133626900.
Oriaku K. A, Directorate of Information Technology, Michael Okpara University of Agriculture, Nigeria., , +2347066003.

booking in advance for six months. In Proposed System user can able to login to the site with valid authentication. If a client wants to book for a train ticket for three months, he can do so by filling all his details in the site, payment and it will be saved in the database. Ticket will be automatically booked when booking open for that particular date and month. So this requirement will help lot of User to access this site. In Our comparative study, we are going to discuss about which model to choose for the project.

- **Deciding Factors**

Before deciding the model to be used, we should get answer for some questions.
1. How stable are the requirements?
2. Who are the end users for the system?
3. What is the size of the project?
4. Where are the project teams located?

III. COMPARATIVE STUDY OF AGILE MODEL VS V-MODEL PROS AND CONS

Agile Model Software Development Model

Agile software development model is a subset of iterative and evolutionary software development methods [17] and is based on iterative enhancement [8] and opportunistic development processes. In all iterative products, each iteration is a self-contained, mini-project with activities that span requirements analysis, design, implementation, and test [17]. Each iteration leads to an iteration release (which may be only an internal release) that integrates all software across the team and is a growing and evolving subset of the final system. The purpose of having short iterations is to enable feedback from iterations N and earlier, and any other new information, can lead to refinement and requirements adaptation for iteration N + 1.

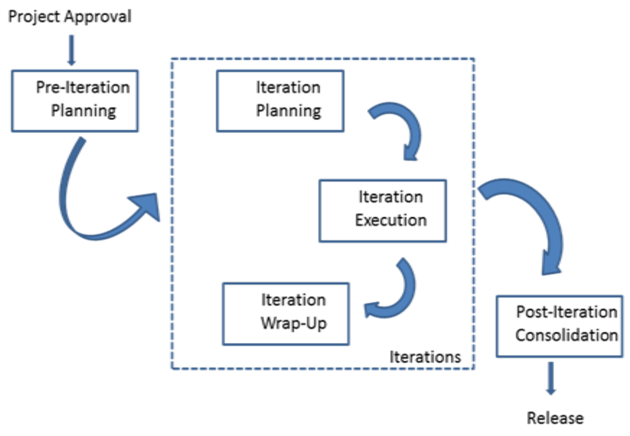

Fig1. Generic Agile Software development Model

In this model, the customer adaptively specifies their requirements for the next release based on observation of the evolving product, rather than speculation at the start of the project [11]. There is quantitative evidence that frequent deadlines reduce the variance of a software process and, thus, may increase its predictability and efficiency.[21] The pre-determined iteration length serves as a timebox for the team. Scope is chosen for each iteration to fill the iteration length. Rather than increase the iteration length to fit the chosen scope, the scope is reduced to fit the iteration length. A key difference between agile methods and past iterative methods is the length of each iteration. In the past, iterations might have been three or six months long. With agile methods, iteration lengths vary between one to four weeks, and intentionally do not exceed 30 days. Research has shown that shorter iterations have lower complexity and risk, better feedback, and higher productivity and success rates [17]. A point of commonality for all agile methods is the recognition of software development as an empirical process.

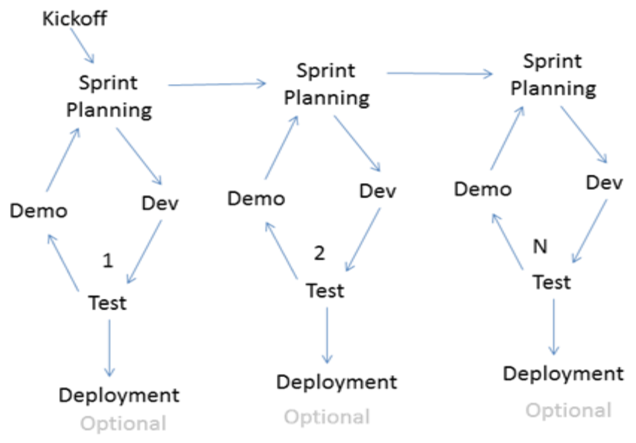

Fig2. Generic Agile Software development Method Cont.

Software development often has too much changes during the time that the team is developing the product to be considered a defined process. A set of predefined steps may not lead to a desirable, predictable outcome because software development is a decidedly human activity: requirements change, technology changes, people are added and taken off the team, and so on. In other words, the process variance is high. Another area of commonality among all agile methodologies is the importance of the people performing the roles and the recognition that, more so than any process or tool, these people are the most influential factoring in any project. Brooks acknowledges the same in *The Mythical Man Month* [9], "The quality of the people on a project, and their organization and management, are more important factors in success than are the tools they use or the technical approaches they take." Unfortunately, there are commonalities among some agile methods that may be less than positive. One is that, unlike more classical iterative methods, explicit quantitative quality measurements and process modeling and metrics are often subdued and sometimes completely avoided.

However, possible justifications for this lack of modeling and metrics range from lack of time, to lack of skills, to intrusiveness, to social reasons. Another potential problem area for agile methods is the ability to cope with corrections or deficiencies introduced into the product. Ideally, even in "classical" development environments, the reaction to change information need be quick; the corrections are applied within the same life-cycle phase in which the information is collected. However, introduction of feedback loops into the software process will depend on the software engineering capabilities of the organization, and the reaction latency will depend on the accuracy of the feedback models. For example, it is unlikely that organizations below the third maturity level on the Software Engineering Institute (SEI) Capability Maturity Model (CMM) scale [18] would have processes that could react to the feedback information in less than one software release cycle. This needs to be taken into account when considering the level and the economics of the “agile” methods. For instance, only relatively small teams can self-organize (one of the agile principles) into something resembling a CMM Level 4 or 5 performance. Also, since personnel resources are not unlimited, there is also some part of the software that may go untested, or may be verified to a lesser degree. The basic, and most difficult aspect of system verification is to decide what must be tested, and what can be left untested, or partially tested [23].

- **Brief History of Agile Model**

The birth of agile software development model could be traced to February 2001, where several software engineering consultants joined forces and began to classify a number of similar change-sensitive methodologies as agile (a term with a decade of use in flexible manufacturing practices which began to be used for software development in the late 1990’s [13]). The term promoted the professed ability for rapid and flexible response to change of the methodologies. The consultants formed the Agile Alliance and wrote The Manifesto for Agile Software Development and the Principles Behind the Agile Manifesto [9]. The methodologies originally embraced by the Agile Alliance were Adaptive Software Development (ASD), Crystal [17], Dynamic Systems Development Method (DSDM), Extreme Programming (XP), Feature Driven Development (FDD) and Scrum.

- **The Principles Of Agile Model**

Customer involvement: Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery: The software is developed in increments with the customer specifying the requirements to be included in each increment. People not process: The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change: Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity: Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

Pros of Agile model

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Cons of Agile model

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

When to use Agile model

- When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the waterfall model in agile model very limited planning is required to get started with the project. Agile assumes that the end users’ needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

Agile Methodologies

Extreme Programming (XP)

The creators of Extreme Programming (XP) aimed at developing a methodology suitable for “object-oriented projects using teams of a dozen or fewer programmers in one location” [29]. Extreme Programming has defined practices and guidelines that implementers should follow. It is an approach to development based on the development and delivery of very small increments of functionality. It relies on constant code improvement, user involvement in the development team and pair wise programming . It can be difficult to keep the interest of customers who are involved in the process. Team members may be unsuited to the intense involvement that characterizes agile methods. Prioritizing changes can be difficult where there are multiple stakeholders. Maintaining simplicity requires extra work. Contracts may be a problem as with other approaches to iterative development.Once completed, the set is tested and put into production. “The goal of each iteration is to put into production some new stories that are tested and ready to go”.

Testing plays a major role in XP. Each iteration is subjected to unit testing. Writing all unit tests prior to writing any code is mandatory. A particular iteration must pass its unit testing prior to going into production. Customers determine system wide tests. Considering their needs and referencing the stories, customers think about what it would take to satisfy them that the iteration is successful. These needs are translated into system wide tests. Testing regularly and often at the unit level and system level provides feedback and confidence that the project is moving ahead and the system is functioning according to the customer's requirements. This process of selecting a set of stories, doing short iterations, working in pairs to code, test, and integrate is repeated until the project is complete. Working in short iterations with constant feedback gives the project the chance to adapt to changing needs. The focus is always on the current iteration. No design work is done in anticipation of future requirements. XP is a highly disciplined process. To be successful, the organization implementing XP must embrace the XP values and principles.

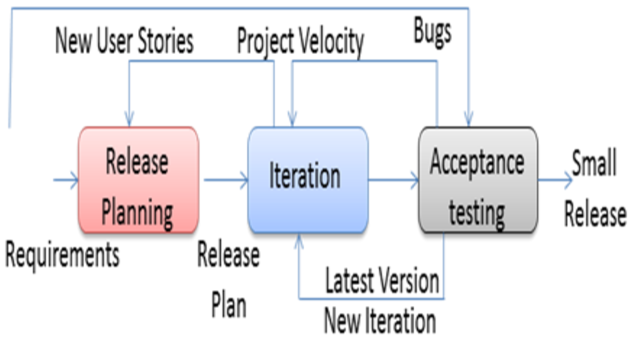

Fig3. Extreme Programming (XP)

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

- **Communication**. XP has a culture of oral communication and its practices are designed to encourage interaction. The communication value is based on the observation that most project difficulties occur because someone *should have* spoken with someone else to clarify a question, collaborate, or obtain help. "Problems with projects can invariably be traced back to somebody not talking to somebody else about something important."
- **Simplicity**. Design the simplest product that meets the customer's needs. An important aspect of the value is to only design and code what is in the current requirements rather than to anticipate and plan for unstated requirements.
- **Feedback**. The development team obtains feedback from the customers at the end of each iteration and external release. This feedback drives the next iteration. Additionally, there are very short design and implementation feedback loops built into the methodology via pair programming and test-driven development.
- **Courage**. The other three values allow the team to have courage in its actions and decision making. For example, the development team might have the courage to resist pressure to make unrealistic commitments.
- **Respect**. Team members need to care about each other and about the project.

Crystal Method

Crystal is a family of processes each applied to different kinds of projects. The idea of having multiple processes stems from the thinking that some projects require fewer rigors than others do. Small and non-critical projects can be developed using less rigorous Crystal methods. Larger and more critical projects, however, demand more attention and therefore, a more rigorous Crystal process is used. Selecting a Crystal process requires that a project be matched to one of four criticality levels.

- Comfort
- Discretionary money
- Essential money
- Life

A system failure for the first level may cause a loss of comfort whereas a system failure for the fourth level may cause a loss of life. Using this as an example a less rigorous process is applied to the former while the latter would demand a highly rigorous process. Each of the processes shares common policy standards:

- Incremental delivery
- Progress tracking by milestones based on software deliveries and major decisions rather than written documents
- Direct user involvement
- Automated regression testing of functionality
- Two user viewings per release
- Workshops for product and methodology-tuning at the beginning and in the middle of each increment.

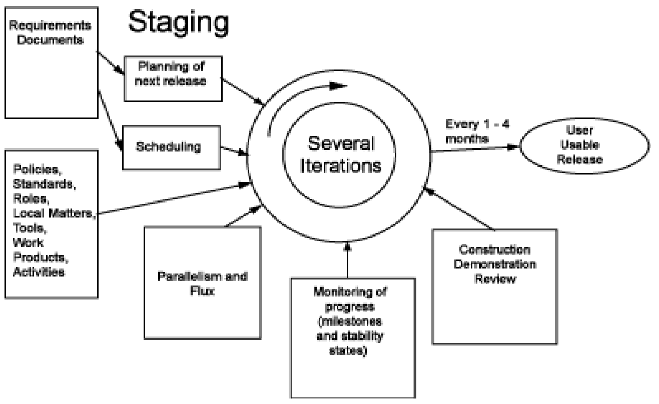
Requirements Documents
Staging
Planning of next release
Scheduling
Several Iterations
Every 1 - 4 months
User Usable Release
Policies, Standards, Roles, Local Matters, Tools, Work Products, Activities
Parallelism and Flux
Monitoring of progress (milestones and stability states)
Construction Demonstration Review

Fig4. Crystal Method of Agile Software Development Model

Scrum Method

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

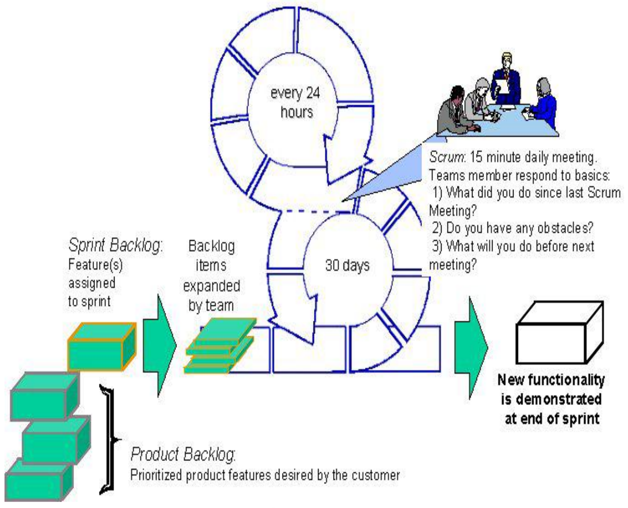
every 24 hours
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
30 days
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

Fig5.The Scrum Process

Problems of Implementing Agile Processes

The most difficult task involved in using agile processes is the ability to integrate them into an organization (i.e. overcoming resistance to existing organizational structures). "Part of the culture is the creation of fiefdoms within the program organization. Adopting agile processes will radically change the functions of the organization within the program and consequently change the staff and funding profiles of the organizations." [13]. The traditional roles played by management, Quality and Assurance, test, financials, and Software Engineers (SWE) will all change creating resistance to the introduction of agile processes. Reading [14] much knowledge is gained concerning the problems experienced by organizations wanting to transition to agile processes. Software Engineers (SWE) are either over zealous or highly skeptical. Over zealous engineers may misinterpret the meaning of agile, taking it to mean "moving quickly" leading toward minimal discipline and turning the project into a hacking free for all. It is important to understand that users of agile processes are making decisions with forethought and reason. On the other hand, there are SWE resisting agile processes because they are strong proponents of design artifacts created using BDUF processes and are most familiar working to a structured plan having a defined time schedule. They do not believe using agile processes produces quality products.

Quality and Assurance (Q and A) and the testing staff often resist agile processes because in the BDUF environment they do not get much attention from management. In the agile process environment, however, this changes because Q and A and testing is a high profile activity occurring after each iteration. Viewing this attention as micromanagement may cause them to resist any change. Management, too, is often skeptical of agile processes. They are uncomfortable with not having Gantt charts and other documents used to manage projects. It is common for management to judge the progress of a project by looking to see if a particular document exists or not. Recall, these are artifacts associated with BDUF processes and do not exist in similar form for agile processes. Another area that causes distress for management is not having a final commitment date of delivery, a bottom line cost, and all features documented.

The V-Shaped Software Development Model

The V- model means Verification and Validation model. Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing of the product is planned in parallel with a corresponding phase of development.The V-model was originally developed from the waterfall software process model. It comprises of four main process phases – requirements, specification, design and Implementation and has a corresponding verification and validation testing phase. Implementation of modules is tested by unit testing, system design is tested by Integration testing, system specifications are tested by system testing and finally, acceptance testing verifies the requirements. The V-model got its name from the timing of the phases. Starting from the requirements, the system is developed one phase at a time until the lowest phase, the implementation phase, is finished. At this stage testing begins, starting from unit testing and moving up one test level at a time until the acceptance testing phase is completed. During development stage the program will be tested at all levels simultaneously.

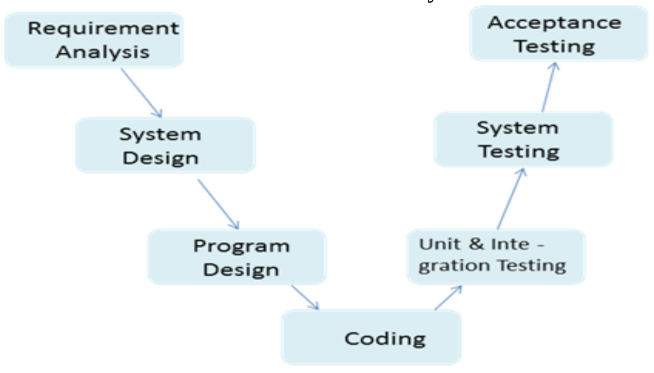

Fig6. The V-Shaped Software Development Model

The different levels of the V-Model are: unit tests, integration tests, system tests and acceptance test. The unit tests and integration tests ensure that the system design is followed in the code. The system and acceptance tests ensure that the system does what the customer wants it to do. The test levels are planned so that each level tests different aspects of the program and so that the testing levels are independent of each other. The traditional V-model states that testing at a higher level is started only when the previous test level is completed.

Pros of V-model

- Simple and easy to use.
- Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
- Proactive defect tracking – that is defects are found at early stage.
- Avoids the downward flow of the defects.
- Works well for small projects where requirements are easily understood.

Cons of V-model

- Very rigid and least flexible.

- Software is developed during the implementation phase, so no early prototypes of the software are produced.
- If any changes happen in midway, then the test documents along with requirement documents has to be updated.

When to use the V-model

- The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
- The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.

IV. CONCLUSION

In this work, we presented an overview of the Agile and V-shape software development models and the characteristics of the projects that they maybe suited to. Additionally, we provided overviews of three sub representative methodologies of agile model, XP, Crystal, and Scrum. As we discussed on Agile model & V-Model; their Pros and Cons, it depends solely upon the organization to choose the model that best fit them. If requirement changes frequently and smaller projects, deliver product in short period of time with skilled resources then we can choose "Agile model ". If requirement changes, larger project, proper validation to take place in each phase, tester to be involved in early stages of development, then we can choose "V-Model" but If requirement is clear, larger project then we choose the oldest method "Waterfall Model".

REFERENCES

[1] W. S. Humphrey, Managing the Software Process (Reading, MA: Addison- Wesley, 1989).

[2] I. Jacobson, G. Booch, and J. Rumbaugh, The Unified Software Development Process (Reading, MA: Addison Wesley Longman, 1999).

[3] P. Kruchten, The Rational Unified Process, 3rd ed. (Reading, MA: Addison-Wesley, 2003).

[4] Software Engineering a Practitioner's Approach; Pressman, Roger S. McGraw Hill; 2001; p 20.

[5] Mass hysteria and the delusion of crowds; Kenny, Michael; itopia Technoparkstr 1 8005 Zurich Switzerland; 2001; page 7.

[6] The Manifesto for Agile Software Development; http://agilemanifesto.org/; last referenced Nov 1, 2002.

[7] M. Aoyama, "Agile Software Process and its Experience," International Conference on Software Engineering, Kyoto, Japan, 1998, pp. 3-12.

[8] V. R. Basili and A. J. Turner, "Iterative Enhancement: A Practical Technique for Software Development," IEEE Transactions on Software Engineering, vol. 1, no. 4, pp. 266 - 270, 1975.

[9] F. P. Brooks, *The Mythical Man-Month, Anniversary Edition*: Addison-Wesley Publishing Company, 1995.

[10] B. Boehm, "Get Ready for Agile Methods, with Care," IEEE Computer, vol. 35, no. 1, pp. 64-69, 2002.

[11] B. Boehm, "A Spiral Model for Software Development and Enhancement," Computer, vol. 21, no. 5, pp. 61-72, May 1988.

[12] B. W. Boehm, Software Engineering Economics. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981.

[13] M. Fowler and J. Highsmith, "The Agile Manifesto," in *Software Development*, August 2001, pp. 28-32.

[14] K. Schwaber and M. Beedle, *Agile Software Development with SCRUM*. Upper Saddle River, NJ: Prentice Hall, 2002.

[15] R. Fairley, Software Engineering Concepts. New York: McGraw-Hill, 1985.

[16] P. Kruchten, The Rational Unified Process: An Introduction, Third ed. Boston: Addison Wesley, 2004.

[17] C. Larman, Agile and Iterative Development: A Manager's Guide. Boston: Addison Wesley, 2004.

[18] M. C. Paulk, B. Curtis, and M. B. Chrisis, "Capability Maturity Model for Software Version 1.1," Software Engineering Institute CMU/SEI-93-TR, February 24, 1993, 1993.

[19] C. Larman and V. Basili, "A History of Iterative and Incremental Development," IEEE Computer, vol. 36, no. 6, pp. 47-56, June 2003.

[20] L. Williams and A. Cockburn, "Special Issue on Agile Methods," IEEE Computer, vol. 36, no. 3, June 2003.

[21] L. Williams and R. Kessler, Pair Programming Illuminated. Reading, Massachusetts: Addson Wesley, 2003.

[22] R. Jeffries, A. Anderson, and C. Hendrickson, *Extreme Programming Installed*. Upper Saddle River, NJ: Addison Wesley, 2001.

[23] M. Vouk and A. T. Rivers, "Construction of Reliable Software in Resource-Constrained Environments," in *Case Studies in Reliability and Maintenance*,

[24]W. R. Blischke and D. N. P. Murthy, Eds. Hoboken, NJ: Wiley-Interscience, John Wiley and Sons, 2003, pp. 205 231.

AUTHORS

Okeke Stephen, Department of Computer Science, College of Physical and Applied Sciences, Michael Okpara University of Agriculture, Nigeria., +2348133626900.

Oriaku K. A, Directorate of Information Technology, Michael Okpara University of Agriculture, Nigeria., , +2347066003