BOEHM-Waterfall Methodology: Issues and Challenges

Alhassan Mohammed Baba
Computer Science Department
Nigerian Defence Academy
Kaduna, Nigeria

Ogwueleka Francisca Nonyelum
Computer Science Department
Nigerian Defence Academy
Kaduna, Nigeria

***Abstract*—The successes of majority of software development ventures are linked to the methods deployed in creating them. There are several models for building various kinds of software projects such as waterfall, spiral, iterative and rapid prototyping. However, Boehm-waterfall model is one type of software engineering methods deployed majorly for large-scale software projects in government and companies. The main idea is early planning to minimize design shortcomings before full-scale development. This paper examined the challenges and issues associated with waterfall model, and appropriate recommendations are made.**

Keywords-Boehm-waterfall; software development; software projects; software engineering; spiral models; iterative models

I. INTRODUCTION

Standish group in 2009 reports revealed that only one-third of software projects could be considered successful [1]. This implies that software projects' failure rate remains unacceptably high, which could be attributed to the increased complexity of software development projects besides the absence or the poorly applied risk management process. In order to achieve project success, we believe that the best way to manage risks in software projects is to select the most suitable methodology that best fits the intended project, and to consider it during the development process as a means to manage risks.

A software engineering methodology or a software development process model is a style to the Software Development Life Cycle (SDLC) that explains the sequence of steps to be followed while developing software projects [2], [3]. According to [2], SDLC models are techniques for designing, building, and maintaining information and industrial systems. Many software development methodologies exist, varying from each other in terms of time to release, quality, and risk management. Regardless of the followed methodology, the basic lifecycle activities are included in all lifecycle models, but probably in different orders. These models might be sequential (waterfall) or iterative (evolutionary). They might be specification-driven (waterfall), code-driven (evolutionary), or risk-driven (spiral). Moreover, they might be conventional (traditional waterfall) or agile (scrum).

Barry Boehm's assertion of software development model is to present a highly evocative and intuitive image of entire project before they are realised [4]. In fact, there is no ideal model that fits all the software development projects; in certain circumstances, each model has its advantages and disadvantages. Deciding upon the methodology to follow depends on the development environment, the type of the project underdevelopment, the development team, and the potential risks. Thus, it falls on the developer to select the methodology (or any customized combination) that best fits the project circumstances [5].

After its creation in the late 1950s, software systems have intensely progressed in terms of size, complexity, presence and importance. Consequently, through this evolution, different issues related to the development of software have emerged. One of the most common critiques is the appreciation about how unpredictable software projects are. Software engineering emerged as a discipline in 1968 at the NATO Software Engineering Conference, and has been review mechanisms to address the challenges of increasing software size and complexity [5].

Efforts have covered a wide range of categories including improvements in programming languages, development techniques, development tools and development methodologies. The waterfall model, one of the first software development methodologies developed in the 1970s, is one of the most remarkable examples of engineering applied to software [6]. One of the most important contributions of this model was the creation of a culture of thinking before coding. In the 1980's, and in the absence of other approaches, this model became a development standard. This model, with some variations, is still widely used in the software industry today [6].

II. REVIEW OF RELATED STUDIES

Waterfall model consists of five stages to be accomplished one after the other in order to a turn out a software product [7]. Bassil [7] identified challenges of software project SDLC such as significant budget overruns, late or suspended deliveries and dissatisfactions of customers. These were attributable to inability of project managers to effectively allocate optimal team members and resources to diverse activities of SDLC. In addition, certain stages of SDLC are idle because of insufficient resources and while others have surplus resources are idle, bringing about shortcoming between arrival and delivery of projects.

Few scholars undertook the comparisons of cost, duration, and software modelling methods. Dash and Dash [8] discussed the waterfall model and its exposure to risks throughout the SDLC. Ruparelia [3] reviewed the most popular software development process models in terms of the application types each fits. Munassar and Govardhan [9] conducted a comparative study between the dominant methodologies, illustrated their phases, advantages and disadvantages, and how they differ from each other. Hijazi et al. [10] identified that the choice of methodology is connected with the several factors, because it helps in determining and assessing risky projects. The author considered several methodologies and the degree to which each methodology supports risk management. The work investigated the state of risk and risk management in the most popular software development process models (that is, waterfall, v-model, incremental development, spiral, and agile development). Munassar and Govardhan [9] examine the area of software development through the development models known as software development life cycle. Authors selected five of the development models namely; waterfall, Iteration, V-shaped, spiral and extreme programming in order to analyse their advantages and disadvantages. The study represents different models of software development and makes a comparison between them to show the features and defects of each model.

III. ANALYSIS OF WATERFALL MODEL

In the past four decades, software has progressed to a complete product from an initial tool used for solving a problem or evaluating information of from problem to solution. Though, the initial programming stages have built-in numerous of problems turning software, which is an impediment to most software development, especially those requiring computers to arrive at solutions. Software comprises of documents and programs collection that have been established to be a part of software engineering procedures. In addition, the goal of software engineering is to generate suitable work plan that builds programs of high quality as shown in Figure 1.

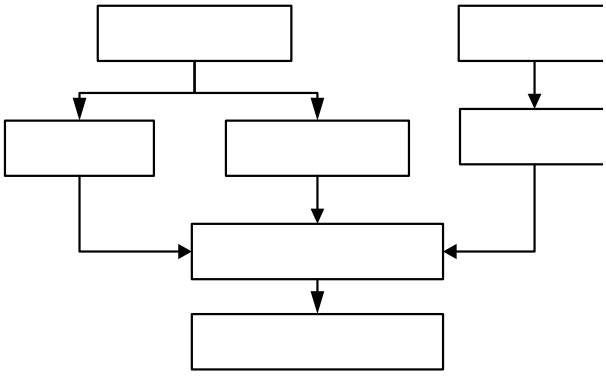

Figure 1. Generic concept of software development process [9].

In Figure 1, the concept can be referred to as abstract denotation of a software process model that encompasses specification, design, validation and evolution. In 1970, Royce was first to introduce water model in an informal style. It abstracts the important software development activities (requirements, analysis, design, coding, testing, and operation) in a sequential manner [11]. Waterfall development was proposed to avoid the risks introduced by the code and fix technique by inserting the requirements and analysis stages before the coding stage. This ensures that user's requirements are clearly defined in advance; as it reduces the time and effort misused on several iterations of code and fix.

In the original waterfall model, any error happening at any stage propagates into the successive stages until it is discovered in the testing phase lately. To avoid this risk, Royce [11] suggested that at the beginning of each stage, a review to the previous stage should be conducted to ensure that the previous stage was properly done. Later, Boehm modified the original waterfall model by adding localized iterations that provide feedback to the preceding phases as shown in Figure 2.

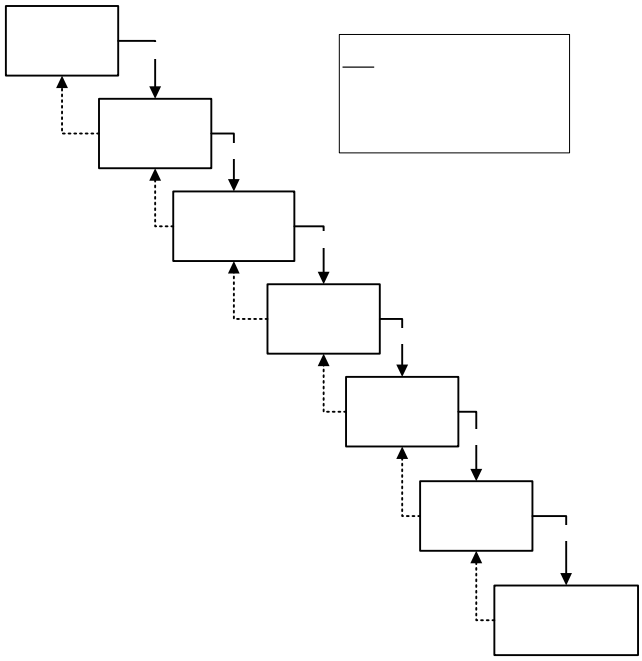

Figure 2. Boehm's waterfall model [12].

Nevertheless, even with these localized iterations, problems are still being revealed in the testing stage, which are usually attributed to glitches in the requirements phase or in the design phase. Thus, to recover from these errors, complex iterations to the requirements phase and to the design phase were added. These iterations consume a lot of time, efforts, and other resources.

In order to avoid the risks of the operational constraints, Royce [11] suggested a preliminary design phase to be interleaved between the requirements phase and analysis phase in order to impose constraints on the analysts. This is accurately accomplished by the iterative loop between the preliminary design and the analysis stages until an adequate preliminary design is attained [10].

One classical model of software engineering is known as waterfall model. This model is one of the oldest models and is widely used in government projects and in many major companies [9]**.** As this model emphasizes planning in early stages, it ensures design flaws before they develop. In addition, its intensive document and planning make it work well for projects in which quality control is a major concern. The pure waterfall lifecycle consists of several non-overlapping stages, as shown in Figure 3.

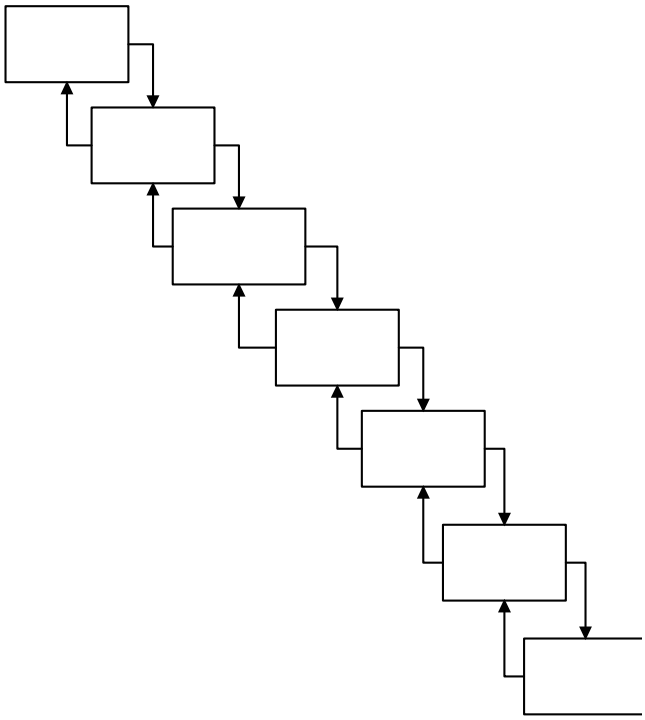

Figure 3. Waterfall software development model [9].

The model begins with establishing system requirements and software requirements and continues with architectural design, detailed design, coding, testing, and maintenance. The waterfall model serves as a baseline for many other lifecycle models as shown in Figure 3 [9].

There are two variants of waterfall models of software development process namely pure and modified [9]. The pure waterfall model performs well for products with clearly understood requirements or when working with well-understood technical tools, architectures and infrastructures as shown in Figure 3. Its weaknesses frequently make it inadvisable when rapid development is needed. In those cases, modified models may be more effective. Modified waterfall model makes use of similar as though the pure waterfall model, except that it is based on a discontinuity approach. It may involve overlapping and splitting of subprojects whenever necessary during architectural and detailed designs as shown in Figure 4.

The strengths over pure model include flexibility, phase continuity for project staff, reduced documentation and ease of implementation. However, ambiguity of milestone is higher; miscommunication for parallel activities are unseen interdependencies are major worry for experts [6].

Figure 4: Pure waterfall model [13].

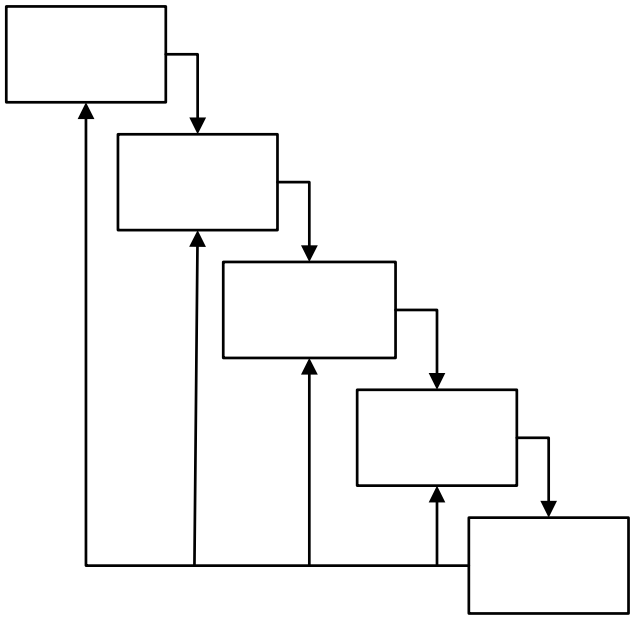

Figure 4: Pure waterfall model [13].

A. Features of Waterfall Model

Aside being a classical model of software engineering, it is one of the ancient and commonly used models in government projects and foremost companies. It underscores early stages planning; it identifies design flaws before embarking on development. More so, it supports thorough documentation and planning which advantageous for quality control issues in software projects. The modelling process kick starts with system and software requirements, progress through the architectural design, detailed design, coding, testing, and maintenance phases.

- **Highpoints:** It is easy to realize and implement. It underpins good practice such as define-before-design, and design-before-code. It ascertains milestones and deliverables, document driven, published documentation standards, effective for established products and inexperience projects teams.
- **Low-points:** The conceptual frameworks of projects may not be realized in real world situations. It is unsupportive of iterative characteristic of exploratory development. More importantly, there is no possibility of attaining precise requirements in the early stages of project lifecycle. Often, delays in detecting serious flaws can result in late project delivery. There is no potential of risk management integration. Changes to projects documentations are largely costly. The costs incurred by this model considering small teams and projects. There is substantial overhead of administration [14].

SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS

B. Description of Software Development Process Phases

The details of common phases of software development process include: [6], [14].

- **System Requirements**: It determines the components for building the system, including the hardware requirements, software tools, and other necessary components. Examples include decisions on hardware, such as plug-in boards (number of channels, acquisition speed, and so on), and decisions on external pieces of software, such as databases or libraries.
- **Software Requirements**: It determines the expectations for software functionality and identifies which system requirements the software affects. Requirements analysis includes determining interaction needed with other applications and databases, performance requirements, user interface requirements, etc.
- **Architectural Design**: This establishes the software framework of a system to meet the specific requirements. This design defines the major components and the interaction of those components, but it does not define the structure of each component. The external interfaces and tools used in the project can be determined by the designer.
- **Detailed Design**: This examines the software components defined in the architectural design stage and produces a specification for how each component is implemented.
- **Coding**: Implements the detailed design specification.
- **Testing**: Determines whether the software meets the specified requirements and finds any errors present in the code.
- **Maintenance:** Addresses problems and enhancement requests after the software releases.

In certain organizations, a change control board maintains the quality of the product by reviewing each change made in the maintenance stage. Consider applying the full waterfall development cycle model when correcting problems or implementing these enhancement requests.

C. Risks Issues and Challenges of Waterfall Models

It is obvious that risks in the waterfall model are unavoidable, even in the Royce's modified waterfall model. This is due to the nature of the model itself which are identified as follows: [11]

- Unbroken change of Requirements: The major risk factor threatens the waterfall projects is the continuous requirements change during the development process. The waterfall model cannot accommodate with these changes due to its strict structure. The waterfall model requires that all requirements be clearly defined in advance in the requirements stage in order to guarantee that no change could appear later on during the development process. Clearly, this is an idealistic situation, since it is difficult for the real projects to identify all requirements previously. Thus, it is even impossible to guard requirements from being changed. Actually, continuous requirements change is not a problem to be solved, neither it is restricted exclusively to the waterfall model. Rather, it is the unstable nature of the software projects besides the highly strict nature of the waterfall model what made its consequences significant in the waterfall model mainly.
- Stages Overlaps are Non-existent: Another source of risk in the waterfall model is that it requires each stage to be completed entirely before proceeding into the subsequent phase. In other words, it does not allow overlapping between stages. Obviously, this will waste time, cost and other resources, since the stages in the waterfall model are relatively long. Hence, most team members who are responsible for specific stages will spend most of their time waiting for other stages to complete so that they can start doing their work.
- Quality Assurance is Poor: Lack of quality assurance during the different phases of the development process is another source of risk. Validating the product is restricted to a single testing phase lately in the development process. Hence, the testing phase in the waterfall model is the highest risky phase, since it is the last stage wherein the system is put as a subject for testing. Thus, all problems, bugs, and risks are discovered too late when the recovering from these problems requires large network, which consumes time, cost, and effort.
- Stages are Long Walkthroughs: Another source of risk in this model resides in the relatively long stages, which makes it difficult to estimate, time, cost, and other resources required to complete each stage successfully. Moreover, in the waterfall model, there is no working product until late in the development process when the product is almost complete and any change is impossible. The worst scenario can be illustrated when the product failed to meet end users' hopes.

D. Potential Solutions to the Problems Identified

- There is need to reduce the number of stages and phases leading up to final product, which can achieve quicker and faster prototyping of engineering products before delivery. This enables product consumers and users to have first-hand experience of project being developed. In addition, incidences of project failures arising from inadequate capturing of users' requirements can be minimized accordingly.
- There is need to layout quality assurance standards and requirements for each phase and stage of product development appropriately. This should include individual stage and unit testing to improve on the software product quality when it was finally delivered. The goal is to reduce errors and time of

delivery attributed to long scheduled phase of debugging and testing.

- The recent advancements in technology have increased the possibility of team members of software projects to collaborate and perform different and similar tasks from different locations at the same time. This can be achieved through overlapping of stages and phases of software product development. Therefore, there is need to have a well-structured layout of activities and tasks for software project throughout all the stages from the commencement to delivery. Such documents guide different team members in working independently from others on similar or different stages of project lifecycle.
- Flexibility is a unique feature of modern software projects. The needs and requirements of end users and consumers continue to change under different experiences and conditions whenever they come across it by rapid prototyping and product visualization methods. The designed can be modified in manner to continue to accommodate fresh requirements and concerns of the product consumers at any levels or stages of product development. The outcome is efficient, less errors, timely and useful product.

IV. CONCLUSION

The research paper has reviewed one of the leading software engineering process models and identified the risks issues and challenges. The papers suggested management of these issues identified with waterfall model. It found that certain software development methods such as waterfall model intrinsically encompass risks management, which implies that risks are inevitable in most software engineering processes, and all software development methodologies, including the risk-driven ones, require that risk management be enhanced in it. An interesting dimension for future research is to find out a strategy that aims at minimizing risk issues and challenges in the different software development methodologies including waterfall. A risk reduction spiral is added to the top of the Waterfall to decrease risks prior to the waterfall phases. The waterfall can be modified using options such as prototyping, Joint Application Designs (JADs) or Cyclic Redundancy Check (CRC) sessions or other techniques of requirements gathering done in overlapping phases.

REFERENCES

[1] Standish Group, "CHAOS Report. 2009," Boston, pp. 1-5, 2009.

[2] L. Guimares, and P. Vilela, "Comparing Software Development Models Using CDM, " in Proceedings of the 6th Conference on Information Technology Education, New Jersey, pp. 339-347, 2005.

[3] N. Ruparelia, (2010). "Software Development Lifecycle Models," ACM SIGSOFT Software Engineering Notes, vol. 35, no. 3, pp. 8-13, 2010.

[4] L. J. Osterweil, "A Process Programmer Looks at the Spiral Model: A Tribute to the Deep Insights of Barry W. Boehm. International Journal of Software Informatics, vol. 5, no. 3, pp. 457-474, 2011.

[5] I. Sommerville, "Software Process Models," in ACM Computing Surveys, vol. 28, no. 1, pp. 269-271, 1996.

[6] I. Sommerville, Software Engineering (9th ed.)," Boston: Addison Wesley, pp. 1-80, 2010.

[7] Y. Bassil, "A Simulation Model for the Waterfall Software Development Life Cycle," International Journal of Engineering and Technology, vol. 2, no. 5, pp. 1-7, 2012.

[8] R. Dash, and R. Dash, "Risk Assessment Techniques for Software Development," European Journal of Scientific Research, vol. 42, no. 4, pp. 629–636, 2010.

[9] N. Munassar, and A. Govardhan, "A Comparison between Five Models of Software Engineering," in International Journal of Computer Science Issues, vol. 7, no. 5, pp. 94–101, 2010.

[10] H. Hijazi, T. Khdour, and A. Alarabeyyat, "A Review of Risk Management in Different Software Development Methodologies," in International Journal of Computer Applications, vol. 45, no. 7, pp. 8-12, 2012.

[11] W. Royce, "Managing the Development of Large Software Systems," in IEEE WESCON, pp. 1-9, 1970.

[12] E. M. Chocano, "Comparative Study of Iterative Prototyping vs. Waterfall Process Applied to Small and Medium Sized Software Projects.," Unpublished M.Eng Thesis, Department of System Design and Management Program, MIT, MA. Retrieved on May 24, 2017 from http://hdl.handle.net/1721.1/34801, 2004.

[13] I. Sommerville, "Software Engineering (7th ed.)," Boston: Addison Wesley, pp. 1-80, 2004.

[14] Bhuvaneswari, T. and S. Prabaharan, "A Survey on Software Development Life Cycle Models," International Journal of Computer Science and Mobile Computing, vol. 2, no. 5, pp. 262-267, 2013.

AUTHORS PROFILE

Alhassan Mohammed Baba holds a B.Tech, M.Tech in Computer Science from Federal University of Technology, Minna, Nigeria. He is currently a PhD Research Student of Department of Computer Science, Nigeria Defence Acadamy, Kaduna, Nigeria. He is a member of Computer Professionals Registration Council of Nigeria (CPN) and Nigerian Computer Society (NCS),

Francisca Nonyelum Ogwueleka is a Professor of Computer Science and current Dean, Faculty of Military Science and Interdisciplinary Studies, Nigerian Defence Academy, Kaduna, Nigeria. She obtained a Bachelor of Engineering (B.Eng) in Computer Science & Engineering, Master of Science (M.Sc) and Doctor of Philosophy (Ph.D) degrees in Computer Science. She is a member of Computer Professionals Registration Council of Nigeria (CPN), Nigerian Computer Society (NCS), Association for Computing Machinery (ACM), Society of Digital Information and Wireless Communications (SDIWC) and International Association of Engineers (IAENG). Although, her research primary domain is data mining techniques, tools and methods; big data and cloud security, she has published standard research work in different aspects of computing and multidisciplinary fields in reputable international journals. Outside of the scope of developmental problems, she has research contributions in the areas of Internet architecture, machine-to-machine applications, digital signal design, image processing and analysis, email classification, intrusion detection, steganography, penetration testing solutions, information retrieval and security. She has supervised several Computer Science PhD dissertations, Masters in Computer Science, Information Technology, Computer Engineering, Cyber Security Science, Information Management Technology and PGD projects in Computer Science. She has supported and collaborated with other Universities and even the Nigerian Universities Commission in curriculum development and capacity building. She has nine (9) published books and over eighty (80) good impact factor international journal publications.

.