African Journal of Science Policy and Innovation Management Vol. 1 (June 2020) 145-157

OVERCOMING THE CHALLENGES OF SOFTWARE DEVELOPMENT IN NIGERIA: THE *SHEWA* MODEL

Gbonjubola O. Binuyo*

African Institute for Science Policy and Innovation, Obafemi Awolowo University, Ile-Ife, Nigeria
*Corresponding author: gobinuyo@gmail.com

ABSTRACT
Software developers employ different software models as tools for achieving software project goals. Software developers in Nigeria face a lot of challenges in this regard due to constant change of requirements. Though literature reveals many different models that can be used to plan, undertake and monitor software development projects, many of them may not be applicable in the Nigerian setting due to technical factors and other anomalies. This paper aims to develop a model for software development tailored towards the Nigerian context. In doing this, software development professionals were contacted. The primary tool for data collection was questionnaire, two hundred and fifty (250) of which were administered in Lagos, Port Harcourt and the Federal Capital Territory, Abuja. Of these, 92% was retrieved and formed the basis for the analyses. The software developers were asked to select key success factors of software development application in the Nigerian milieu from four purposively selected models which are waterfall, spiral, incremental, and prototyping from literature. Subsequently, a flow chart diagram was sketched to consolidate the key success factors to develop a model. The model, named Shewa, was tested using internal factors such as licencing and linkages (4.43), human resource (3.69) and external factors such as access to technical information and support (3.65) and competition with international market (3.59) among others. The results show that Shewa is applicable in Nigeria given the internal and external factors.

Keywords: Software Development; Models, Stakeholders; Challenges; Shewa; Nigeria.

1. Introduction

The computer software and services industry is a key example of a knowledge production sector. Software is the interface between computers and the humans who use them. Software development reflects the capacity to respond rapidly to changes in products and processes with opportunities for innovation and technological capability (Binuyo, 2012). The value of what a software company produces is almost entirely in the knowledge embodied in its products and services. Software development is a fast growing industry, producing high value products and services that can provide an edge in competing with developed countries and bring about comparative advantages (Tatikonda and Rosental, 2000). The software industry is dominated by firms based in developed countries. However, a strong domestic software industry will offer great prospects for economic growth and industrial development in developing economies.

Attempts to ensure sustainable economic development and poverty reduction of most nations usually involve the development of the agriculture, mining, industrial and the software service sectors (Hoda *et al*., 2010). The Industrial Revolutions in Europe and the United States of America have been premised on technological innovation and breakthroughs (Gaio and

Brazilian, 1998). During the late 1990s, Information and Communications Technology (ICT) was the largest contributor to growth within capital services for both Canada and the United States (Harchaoui *et al*., 2002). Similar trends have been observed with economic development in China, Korea, Taiwan, India, South Africa, and other emerging economies (Pressman, 1999). In 2000, the Federal Government of Nigeria embarked on an aggressive drive towards the provision of more efficient services in the Nation through its privatisation and deregulation policies. This initiative led to the development of the National Information and Communications Technology Policy in December 2001. The policy, among other things, recognised the need for the establishment of an enabling environment for deregulation and rapid expansion of Information and Communications services in the country. The mission statement of the government was to use ICT as a tool for education, creation of wealth, poverty eradication, job creation, and global competitiveness (Kent, 2000). The policy objective was to develop globally competitive manpower in ICTs and related disciplines. This entailed the development of a pool of ICT engineers, scientists, technicians and software developers among others. Consequently, attractive career opportunities were expected to emerge in addition to the development of made-in-Nigeria software and other computer components that can earn the Nation foreign exchange.

The spectacular growth of the software industry in some non-G7 economies has aroused both interest and concern (Alan, 2016). For example, few areas of production, engineering and education do not include software as an important and increasingly complex component (Pekka *et al*., 2009). Moreover, new small firms with relatively few tangible assets can still prosper and grow rapidly. It was discovered that the software industry has the potential to become one of the most internationally dispersed high-tech industries (Alistair, 2004). In the last two decades, there has been high growth rate in this sector and a dramatic increase in the adoption of computer software and services worldwide, with a tremendously high level of productivity (Hullet, 2001, Hermann, 2012). Therefore, governments all over the world are struggling with the problem of how to ensure that Science, Technology and Innovation (STI) contribute effectively to solving national problems.

Technology per se does not solve social problems. But the availability and use of ICT, and software development in particular, are a pre-requisite for economic and social development. Similarly, the crucial role of software development as a pivot that drives ICT in facilitating development is two-pronged (Andrew and Nachiappan, 2007). First, it allows countries to leapfrog stages of economic growth by enabling them to modernize their production systems and increase their competitiveness faster than in the past. Also, ICT is an essential tool for economic development and material well-being (Pekka *et al*., 2003). Furthermore, studies have shown close relationships between the diffusion of information technology, software development, productivity and competitiveness (Gregory *et al*., 2015). An adequate level of education in particular, is essential for the design and productive use of software development processes and products as critical tools to drive new technologies (Jim and Alistair, 2010; Versionore, 2016). In spite of the global relevance of software development as a strategic tool that facilitates growth and development, it is not similarly deployed across all countries (Mikecohn, 2010). Specifically, the software development sector in some developing countries like Nigeria seems not to have kept pace with the dynamic nature of the sector and with global trends.

Binuyo (2012) reported that the challenges of the Nigerian software industry are varied and multi-dimensional. These include but are not limited to lack of infrastructure for software development, dearth of skilled manpower, and knowledge incubation facilities. Other challenges identified are low patronage of indigenous software products, stiff competition from foreign

software products, and more importantly, the lack of continuity of indigenous software development efforts (Kai *et al*., 2009). This has resulted in most software development projects being done haphazardly or abandoned without the documentation of lessons learned. This lack of knowledge incubation on software development in Nigeria has stifled rather than stimulated rapid development in the sector (Arie, 2001; Anderson and Carmichael, 2016).

In view of the above mentioned benefits of software development and challenges in developing them, there are many models in the literature which can be used to plan, undertake, better deliver and monitor software development projects. Anderson and Carmichael (2016) and Laurel (2010) described various types of software developing life cycles (SDLC) models or methodologies are Waterfall Model, V-Shaped Model, Evolutionary Prototyping Model, Spiral Method (SDM), Iterative and Incremental Method, and Agile development. However, applying these existing models in developing countries such as Nigeria comes with some weaknesses. The aims of this paper is to examine the weaknesses and advantages of selected models for software development projects from the view of certain categories of experts involved in software development in Nigeria, and thereafter develop a new model based on the outcome. It is hoped that this new model may be useful for practice and policy. This section is followed by the literature review, methodology, result, discussions and conclusion.

2. Literature Review

2.1. *Stages in software development (Software Development Lifecycle (SDLC))*

There are different approaches to software development. Some take a more structured, or engineering-based approach to developing business solutions, whereas others may take a more incremental approach, where software evolves as it is developed piece by piece (Oumer *et al*., 2016). Most models or methodologies share some combination of the following stages of software development: market research, gathering requirements for the proposed business solution, analysing the problem, devising a plan or design for the software-based solution, implementation (coding) of the software, testing the software, deployment, maintenance and bug fixing. These stages are often referred to collectively as the software development life-cycle (SDLC) (Bandalos and Boehm-kanfam, 2009; Macro *et al*., 2017).

The different approaches to software development may necessitate carrying out these stages in different order, or devote more or less time to different stages. The level of detail of the documentation produced at each stage of software development may vary. These stages may also be carried out in turn, or may be repeated over various cycles or iterations. The SDLC usually involves less time spent on planning and documentation, and more time spent on coding and development of automated tests (Freeman and Louca, 2002; Petersen and Claes, 2009). Other approaches also promote continuous testing throughout the development lifecycle, as well as having a working (or bug-free) product at all times. More structured approaches attempt to assess the majority of risks and develop a detailed plan for the software before implementation (coding) begins. These approaches avoid significant design changes and re-coding in later stages of the software development lifecycle (Bowen and Starvridou, 2007). Various software development models are discussed in the following section.

2.2. Software development models

Every software development methodology has more or less its identifiable approach. There is a set of more general approaches, which are developed into several specific methodologies. These approaches (Doz *et al*., 2002; Mikecohn, 2010) are waterfall (linear framework type); prototyping (iterative framework type); incremental (combination of linear and iterative

framework type); spiral (combination linear and iterative framework type); and Rapid Application Development (RAD), among others.

Waterfall model

The waterfall model is a sequential development process, in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirement, analysis, design, implementation, testing/validation, integration, and maintenance (Doz *et al*., 2002). Basic principles of the waterfall model are:-

(i) project is divided into sequential phases with some overlap. Splash backs are acceptable between phases.

(ii) emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time (Blackburn *et al*., 1998; Petersen and Claes, 2009).

Tight control is maintained over the life of the project through the use of extensive written documentation, as well as through formal reviews and approval/signoff by the user. Information technology management occurs at the end of most phases before beginning the next phase.

Prototyping

This describes the framework of activities during which incomplete versions of the software program is developed. Basic principles of prototyping (Boehm, 1998; Ayelt *et al*., 2017) are:

i. not a stand-alone complete development methodology, but rather an approach for handling selected portions of a larger, more traditional development methodology (i.e. Incremental, Spiral, or Rapid Application Development (RAD).
ii. it is an attempt to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
iii. the user is involved throughout the process, which increases the likelihood of user acceptance of the final implementation.
iv. small-scale mock-ups of the system are developed following an iterative modification process until the prototype evolves to meet the users' requirements.
v. most prototypes are developed with the expectation that they will be discarded. It is however possible in some cases to evolve from a prototype to a working system.
vi. a basic understanding of the fundamental business problem is necessary to avoid solving the wrong problem.

Incremental

Various methods are acceptable for combining linear and iterative systems development methodologies, with the primary objective of each being to reduce inherent project risk. The incremental method breaks a project into smaller segments and provide more ease-of-change during the development process. The basic principles of incremental development (Badawy, 2008) are:

(i) A series of mini-Waterfalls are performed, where all phases of the Waterfall development model are completed for a small part of the systems before proceeding to the next increment, or

(ii) The overall requirements are defined before proceeding to evolutionary, mini-Waterfall development of individual increments of the system, or

(iii) The initial software concept, requirements analysis, and design of architecture and system core are defined using the Waterfall approach, followed by iterative Prototyping, which culminates in installation of the final prototype (i.e., working system).

Spiral

The spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. The basic principles are as follows:

i. focus is on risk assessment and on minimizing project risk by breaking a project into smaller segments and providing more ease-of-change during the development process, as well as providing the opportunity to evaluate risks and weigh consideration of project continuation throughout the life cycle (Schware, 2000).
ii. each cycle involves a progression through the same sequence of steps, for each portion of the product and for each of its levels of elaboration, from an overall concept-of-operation document down to the coding of each individual program (Torgeir *et al*., 2012).
iii. each trip around the spiral traverses four basic quadrants: (a) determine objectives, alternatives, and constraint of the iteration; (b) Evaluate alternatives; identify and resolve risks; (c) develop and verify deliverables from the iteration; and (d) plan the next iteration (Jim, 2013).
iv. the model begins each cycle with an identification of stakeholders and their win conditions, and end each cycle with review and commitment (Binuyo, 2012).

There are significant advantages and disadvantages associated with the various models or methodologies, and the best approach to solving a problem using software will often depend on the type of problem, aim and goal. If the problem is well understood and a solution can be effectively planned out ahead of time, a waterfall-based approach may work the best (Kruchten *et al*., 2012). If, on the other hand, the problem is unique (at least to the development team) and the structure of the software solution cannot be easily envisioned, then an incremental approach may work best.

The software development process is a structure imposed on the development of a software product. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process (Hullet, 2001).

2.3. ***Sources of ideas for software development and its relevance to human, social and economic development***

The need to develop software and the sources of ideas for software products often emanate from many sources (Doz *et al*., 2002). These ideas can come from market research including the demographic survey of potential new customers; existing customers; sales prospects who rejected the product; relevance of the product to economic, social and human development; other internal software development staff; or a creative third party (Macro, 2017). Ideas for software products are usually first evaluated by marketing personnel for economic feasibility and viability for fit with existing channels of distribution, for possible effects on existing product lines, required features, and for fit with the company's marketing objectives. In a marketing evaluation phase, the cost and time assumptions are evaluated. A decision is reached early in the first phase as to whether, based on the more detailed information generated by the marketing and development staff, the project should be pursued further (Poppendieck and Poppendieck, 2003). In essence, software product planning is critical to development success and absolutely requires knowledge of multiple disciplines (Blackburn *et al*., 1998).

However, software development may involve compromising or going beyond what is required by the client. A software development project may stray into less technical concerns such as human resources, risk management, intellectual property, budgeting, crisis management, etc.

These processes may also cause the role of business development to overlap with software development (Badawy, 2008).

3. Method

Software is meant to be everywhere and is required to operate on a massive and always evolving technology landscape. In order to develop computer software for the evaluation of effectiveness and efficiency of various approaches used, a flow chart was designed to write the algorithm. Specific requirements or conditions that describe all inputs are detailed so that it is made easier to design the model and validate the software according to requirements. It is also important that all the outputs are listed to better define interaction with other systems and facilitate integration. The functionalities and performance criteria need were defined as well. Two sets of questionnaire were developed and used to elicit information on the four selected models (waterfall, spiral, prototyping and incremental). These models have weaknesses and strengths, both of which were considered. These sets of questionnaire were administered on 250 software developers chosen from Goldstar Directory (2007/2008). These respondents were asked to rate (on a 5-point scale ranging from 1 = no influence to 5 = very high influence) the various attributes of these models. The response rate was 92%. From this rating it was found that there are transitions from one model to others with a view to achieving software project's aims and objectives (Pikkarainen *et al*., 2008). The major weaknesses of the selected models were considered to develop the *Shewa* model for the study as depicted in (Table 1). These were combined with both internal and external factors in (Table 2). Any of these approaches, namely the Waterfall, Prototyping, Incremental, Spiral and Shewa, can be applied for this model either singly or interchangeably. Figure 1 is the algorithm of *Shewa* model. It shows that *Shewa* is an iterative model where the existing processes such as waterfall, incremental, spiral, prototyping among others were considered. This algorithm also describes the functionality of the *Shewa* model. The model gives room for modification of the process, creates access for feedback from the customer, determines the effectiveness of the product, gives reasons for non-acceptability of the system if it is not functioning, and re-cycles the system as required. The output (product) is verified to test the approaches adopted and modify the processes involved where necessary. The algorithm was developed to test the new system/model using the data collected on internal and external factors from the field survey as an input.

4. Results and Discussion

1.1. Model of software design (*Shewa* Model)

The model named *Shewa* (Figure 2), is built in parts. The process includes system requirements that allow for review and modification where necessary. It is assessable and it is an inter-related component of the mother design. It gives room for review, valuation and re-evaluation. It is flexible and allows for easy maintenance. Figure 3 shows the effects of internal and external factors on the model employed to develop the software as it might be required by the customer. The graph depicts the output of *Shewa* model after test and compared with the existing approaches examined in this study. This supports Leo and Dan (2008) that the main purpose of software testing is to detect errors and ascertain under which condition(s) it will work.

It was found that internal factors such as human resources, R&D activities, ownership structures, working experience and external factors which include competition, economic, social, access to skill and access to technical information and support were some of the critical factors that are influencing software development. And hence it has positive impact on indigenous software development and thus will enable them to compete globally with their foreign counterparts if

properly applied. The functionality of the adopted approach is shown graphically in Figure 3 which confirms the effectiveness of its usage.

In this environment, individuals and interactions are valued over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan. *Shewa* methods and practices aim to simplify the software process by avoiding "bureaucracy" and advocate short time cycles, close involvement of the client and also an adaptive rather than predictive strategy

1.2. The comparison of Shewa model and the selected models considered in the study

Specifically, the *Shewa* model was designed in cognisance of the weaknesses, advantages and some negative tendencies in other models or approaches considered in this study such as waterfall, incremental, prototyping and spiral. These approaches cannot be built in bits like *Shewa* model. The waterfall approach entails traditional sequential development processes and full knowledge of the system. However, it neither gives room for any expansion nor modification at any stage of the development process. Incremental approach allows upgrading of the system. It is cost effective, allows extensibility of the system and future expansion but the system requirements and resources are broken into many blocks which make it difficult to control. The prototyping approach allows for proper studying of various stages of the system as each stage is independent of one another. It consumes more resources and no rework is allowed at any stage of the development. Spiral and prototype are built in chunks. It is very flexible, allows upgrading at any stage but consumes more time and resources.

Shewa model is not susceptible to security breaches (i.e. it is a hack-resilient software model). It has less likelihood of memory corruption and is thus less susceptible to overflow attacks. Its requirement entails quality of critical systems to solve a defined goal by the end-users. It discovers problems at the early stage of development. It can accommodate in-built security, which cannot be easily circumvented or hacked if implemented securely.

Table 1: Selected approaches compared with "*Shewa* model" for software development

Software Approach	Weakness	Advantages	Implication	Remarks
Waterfall	Require full knowledge of the system No change is allowed	The process of development is faster if all the requirement are known	Any company that uses waterfall approach to develop its software cannot use it for any expansion programme/project.	It cannot accommodate any modification at any of the stages of development
Incremental	System requirements and resources are broken down into too many blocks which makes it difficult to manage	The development process is faster than waterfall It allows new features to be added easily	Upgrading of the system is possible and cost effective	It allows extensibility of the system and future expansion to accommodate new programmes and projects.
Prototyping	The system is neither linear nor integrated.	It allows proper studying of various stages of	Each unit stands on its own and has no link to one another.	It consumes more resources (time, labour, and cost)

	But every stage is undertaken separately as a "job" that must be completed before going to the next stage.	the system as each stage is independent of one another.		because no rework is allowed at any stage of the development process until the process is complete.
Spiral	The prototyping is built in chunks (bit) which consumes more time and resources than others	Extensible. It can accommodate expansion of work	Very flexible for upgrading.	Allow upgrading of the system at any stage of development and at usage levels.

Source: Author's Concept

Table 2: External and Internal Factors influencing Software Development in the Nigerian ICT industry

Factors	Waterfall	Prototyping	Incremental	Spiral	*Shewa*
Licensing and Linkages	4.43	4.40	4.41	4.39	4.49
Human Resources	3.69	3.64	3.59	3.66	3.71
On the job training	3.58	3.56	3.54	3.48	3.68
R&D Activities	3.48	3.35	3.46	3.38	3.57
Availability of Funds for R&D	3.38	3.36	3.35	3.34	3.40
Working Experience	3.37	3.25	3.32	3.27	3.39
Availability of Funds for training	3.21	3.19	3.20	3.18	3.23
Ownership structure	3.31	3.30	3.29	3.32	3.35
Size of Firm	3.02	3.00	2.92	2.88	3.22
Access to technical information &support	3.65	3.63	3. 58	3.60	3.67
Competition with international market	3.59	3.57	3.54	3.50	3.61
Economic factors					
Access to professional skills	3.46	3.42	3. 41	3.39	3.50
from labour market	3.39	3.37	3.34	3.30	3.42
Government support policy	3.35	3.33	3.30	3.27	3.40
Social factors					
	3.30	3.29	3.27	3.25	3.35

Key: No influence = 1, Little influence = 2, Moderate influence = 3, High influence = 4 and very high influence = 5

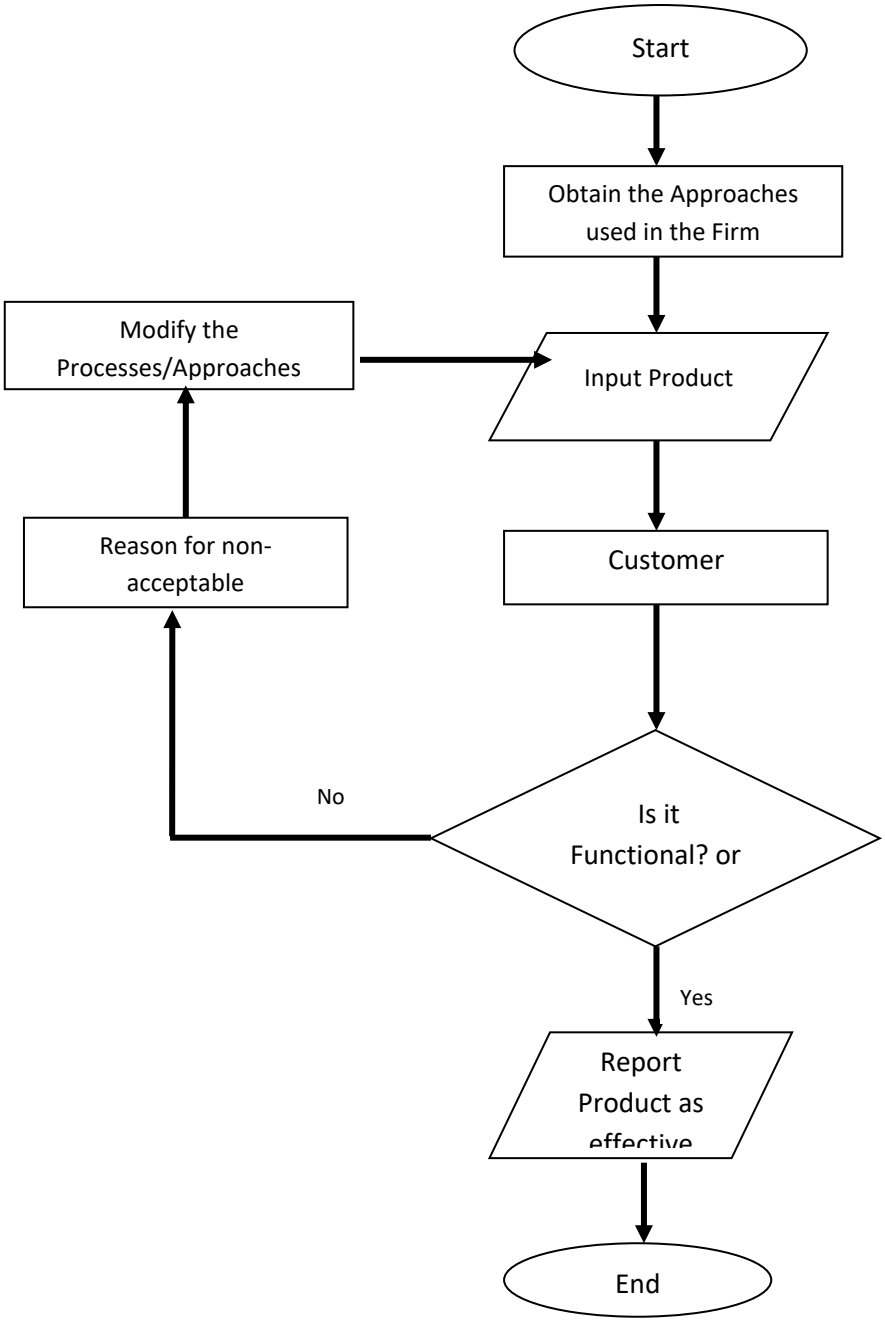
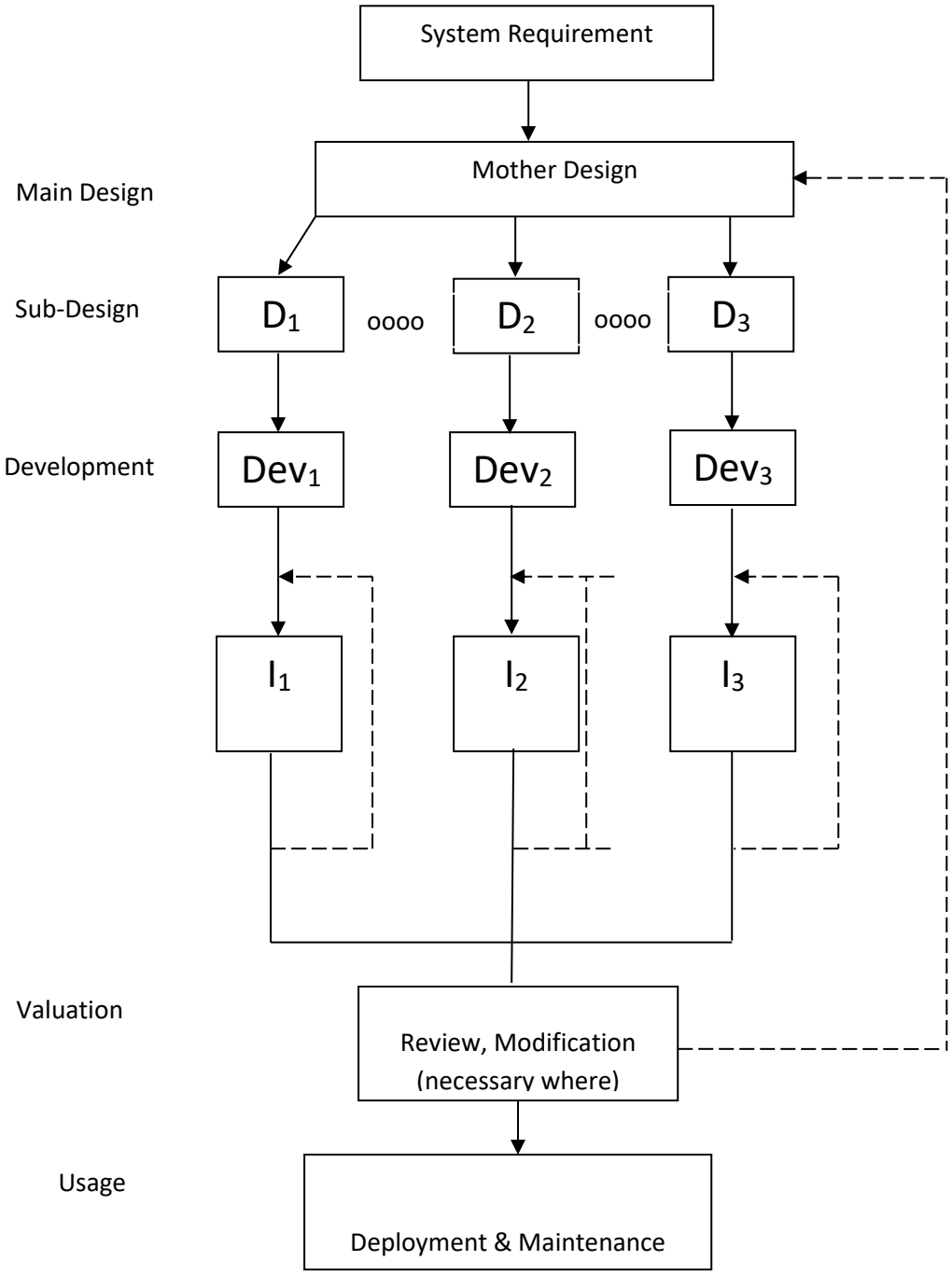


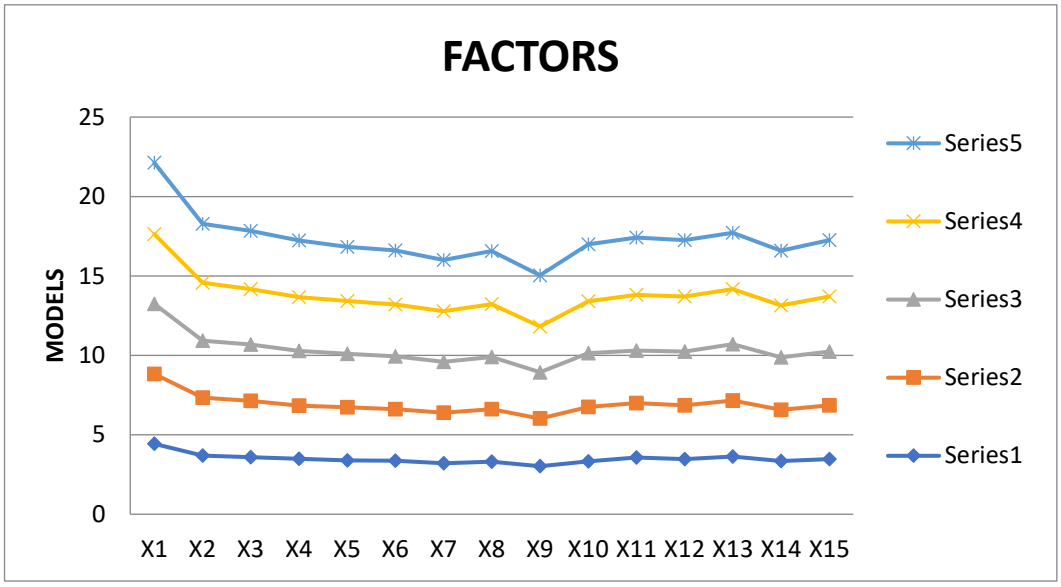
Fig. 1: An Algorithm of *Shewa* model



Key: - - - -The loop is meant for checking or re-working any troubleshooting issue

This represents Inter-related Components of Mother Design

Figure 2: *Shewa* Model Designed on Evaluation of CSWD in Nigeria



Key:
(1) 0-5 = No influence; (2) 6-10 = Little influence; (3) 11-15 = Moderate Influence; (4) 16-20 = High Influence; (5) 21-25 = Very high influence
X1 = Licensing and Linkages; X2 = Human Resources; X3 = On the job training; X4 = R&D Activities, X5 = Availability of Funds for R&D; X6 = Working Experience; X7 =Availability of Funds for training; X8 = Ownership structure; X9 = Size of Firm; X10 = Access to technical information &support; X11 = Competition with international market; X12 = Economic factors; X13 = Access to professional skills from labour market; X14 = Government support policy; X15 = Social factors
Series 1: Waterfall; Series 2: Prototyping; Series 3: Incremental; Series 4: Spiral; Series 5, Shewa

Figure 3: The output of software designed from the project using *Shewa* Model

5. Conclusion

The Software development process has grown widely from the 1970s to date. The way software developers approach development has changed considerably. It is observed that more and more emphasis has been put on creating more user-friendly programs with minimum risk, types of tasks, time and budget required for this new class of software systems. This study provides an overview of the evolution of the new software development approaches. It provides some definitions that are sometimes vague in literature such as what a software process actually is. A summary is provided about weaknesses of some selected approaches that have led to the proposal of the *Shewa* methodology. Strengths and weaknesses of those approaches and *Shewa* processes are surveyed, allowing us to understand the challenges faced when applying one or the other approaches. Therefore, there is poor published evidence about persistent difficulties or failure situations in practice that would allow us to learn from errors. There is no unique way to develop software. The best way depends on specific factors relating the context of the organisation, the project and the development team.Each methodology discussed is unique to the features it provides to the developer and each has its downside which has led to newer methodologies being created. But the concept of all the methodologies is similar, starting with identification, analysis, design,coding, testing, implementation and maintenance. In a nutshell it

is very vital for programmers to follow these methodologies as they provide a framework on which they can build outstanding software that will cater for the needs of their customers. In addition, the developed model approaches present opportunities for "leapfrog" strategies that could accelerate the development of the continent.

Acknowledgement
I acknowledge with gratitude the invaluable support received all through the duration of the research.

REFERENCES

Alan, M. (2016) Managing Agile. Springer, 2016.

Alistair, C. (2004) Crystal clear: a human-powered methodology for small teams. Pearson Education

Anderson, D. J. and Carmichael, J. (2016) Essential Kanban Condensed. Blue Hole Press.

Andrew, B and Nachiappan, N. (2007) Usage and perceptions of agile software development in an industrial context: An exploratory study. In Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on, pages 255–264. IEEE,

Arie, V. D. (2001) Customer involvement in extreme programming. ACM SIGSOFT Software Engineering Notes, 26(6):70.

Ayelt, T., Oliver, L., Eckhart, H., and Christian, R. P. K. (2017) Study report: Status quo agile 2016/2017. Technical report, Hochschule Koblenz University of Applied Sciences, http://www.status-quo-agile.de/.

Badawy, M.K, (2008) Technology Management Education: alternative Models California Management Review, 40 (4), pp. 94 – 115.

Bandalos, T. and Boehm-Kanfman, P, (2009) "Training and Productivity in African Manufacturing Enterprises." Paper presented at the Conference on Enterprise Training Strategies and Productivity. Private Sector Development Department, World Bank (Washington, DC).

Binuyo, G.O., (2012), "Evaluation of Computer Software Development in Nigeria" Unpublished PhD Thesis, African Institute for Science Policy and Innovation, ObafemiAwolowo University, Ile-Ife, Nigeria.

Blackburn, J. D., Scudder, G. D. and Van Wassenhove, L. N, (1998) "Improving Speed and Productivity of Software Development: A Global Survey of Software Developers." IEEE Transactions on Software Engineering. 22(12), pp: 875-885.

Boehm, B.W. (1998) *Software Risk Management,* Washington, *D.C.:* IEEE Computer Society Press).

Bowen, J. and Stavridou, V, (2007) "*Safety-critical systems, formal methods and standards "Software Engineering Journal,* 8 (4), pp. 189-209.

Doz, Y., Santos, J., and Williamson, P. (2002) 'How Companies Win in the Knowledge Economy' in From Global to Multinational. Harvard Business School Press, 2002. Research Technology Management. March – April, 45, (2), 2-8.

Freeman, C. and Louca, F. (2002) As Time Goes By. From the Industrial Revolution to the Information Revolution (Oxford, New York: OUP).

Gaio, F. and Brazilian, G, K (1998) "Software strategies for developing countries: lessons from the international and "Indian groups face struggle to protect their software arms", *Financial Times,* 28 August.

Gregory, P., Leonor, B., Katie, T., Dina, S. and Helen, S. (2015) Agile challenges in practice: a thematic analysis. In International Conference on Agile Software Development, pages 64–80. Springer.

Harchaoui, T. M, F. Tarkhani, C. Jackson, & P. Armstrong (2002), "Information Technology and Economic Growth in Canada and the U.S.", Monthly Labor Review, October.

Hoda, P., Kruchten, P., Noble, J. and Marshall, S. (2010) Agility in context. ACM Sigplan Notices, 45(10):74–88.

Hullet, D.T, (2001) "Key Characteristics of a Mature Risk Management Process", Fourth European Project Management Conference, PMI Europe.

Jim, H, and Alistair, C. (2001) Agile software development: The business of innovation. Computer, 34(9):120–127.

Jim, H (2013) Adaptive software development: a collaborative approach to managing complex systems. Addison-Wesley.

Kai, P., Claes, W. and Dejan, B. (2009) The waterfall model in large-scale development. In PROFES, pages 386–400. Springer.

Kent Beck. (2000) Extreme programming explained: embrace change. Addison-Wesley Professional

Kruchten, P., Robert, L. N. and Ipek, O. (2012) Technical debt: From metaphor to theory and practice. Ieee software, 29(6):18–21.

Laurie, W. (2010) Agile software development methodologies and practices. Advances in Computers, 80:1–44.

Leo, R. V. and Dan Turk. (2008) Agile software development: A survey of early adopters. Journal of Information Technology Management, 19(2):1–8.

Marco, K., Philipp, D., J¨urgen, M., Paolo, T., Vahid, G., Michael, F., Kitija, T., Fergal, M., Oliver, L., Eckhart, H., and Christian, R. P. (2017) Hybrid software and system development in practice: Waterfall, scrum, and beyond. In Proceedings of the 2017 International Conference on Software and System Process, ICSSP, pages 30–39, New York, NY, USA, 2017. ACM.

Michael, H. DevOps for developers. A press, 2012.

MikeCohn. Succeeding with agile: software development using Scrum. Pearson Education, 2010.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P. and Still, J. (2008) The impact of agile practices on communication in software development. Empirical Software Engineering, 13(3):303– 337.

Pekka, A., Juhani, W., Mikko, T. S., and Jussi, R. (2003) New directions on agile methods: a comparative analysis. In Software Engineering, 2003. Proceedings. 25th International Conference on, pages 244–254. IEEE.

Pekka, A., Kieran, C. and Xiaofeng, K. (2009) 'Lots done, more to do': the current state of agile systems development research.

Petersen, K. and Claes, W. (2009) A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. Journal of systems and software, 82(9):1479– 1490.

Poppendieck, M. and Poppendieck, T. (2003) Lean software development: an agile toolkit, 2003.

Pressman, R. S. (1999) A Manager's Guide to Software Engineering, (New York: MsGraw- Hill).

Qumer, A., Brian, G., Sellers, H. and Niazi, M. (2016) Scaling for agility: A reference model for hybrid traditional-agile software development methodologies. Information Systems Frontiers, pages 1–27.

Schware, R., (2000) "Software Industry Entry Strategies for Developing Countries: A "Walking on Two Legs" Proposition", World Development, 20(2). Pergamon Press, pp. 143 – 164.

Tatikonda, M. V, and Rosenthal, S.R, (2000)"Successful Execution of Product Development Projects Balancing Firmness and Flexibility in the Innovation Process." Journal of Operations Management. 18(4), pp. 401-426.

Torgeir, D., Sridhar, N., VenuGopal, B. and Nils, B. M. (2012) A decade of agile methodologies: Towards explaining agile software development.

VersionOne (2016) 11th annual state of agile report. Technical report, Technical report, VersionOne, http://stateofagile.versionone.com/.