IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 1, Ver. III (Jan.-Feb. 2017), PP 42-52
www.iosrjournals.org

Assessment of the Approaches Used in Indigenous Software Products Development in Nigeria

G.O. Binuyo[1] *M. O. Alimi[2]
[1]*African Institute For Science Policy And Innovation,Obafemi Awolowo University, Ile-Ife, Nigeria.*
[2] *Computer Science Unit, Physical Sciences Department, Al-Hikmah University, Ilorin, Nigeria*

Abstract: *Acceptability of indigenous software is always low in most of the African countries especially in Nigeria. This paper then study and presents the results on the assessment of the approaches used in indigenous software development products. The study involved ICT firms that specialized in software development and educational institutions who were part of major stakeholders as well as users of software packages. The primary tool for data collection was questionnaire, which was used to elicit information from software developers on the various approaches adopted in their operations. This was also complemented with information from secondary sources. The identified approaches were measured on a five-point Likert scale rating of 5 to 1 to determine their relative strength index (RSI) in the factors. The result revealed the various approaches adopted for software development had significant difference of chi (45)1699.06 at p≤ 0.001 with spiral (6.02), agile (5.86), prototyping (5.67), object oriented (5.48), rotational unified process (5.32), computer and incremental case (4.50), waterfall (3.66) and integrated (1.98) were the commonly adopted approaches used for software development. Similarly, the approaches adopted by software development firms were correlated and returned a significant difference of (Z = 1699.06, p≤ 0.001). The result implies that these approaches had a great impact on the domestic use of software products and perhaps is the most important driver of software industry growth for emerging technologies.*

Keywords: *Software Development, Software products, approaches, stakeholders, indigenous, Nigeria.*

I. Introduction

The computer software and services industry is a key example of knowledge production. Software development reflects the capacity to respond rapidly to changes in products and processes with opportunities for innovation. The value of what a software company produces is almost entirely in the knowledge embodied in its products and services. It is a fast growing industry producing high value variable products and services that could provide the cutting edge in competing with developed countries and bring about comparative advantage in the sector [29]. Software industry is also dominated predominantly by firms based in major industrialized countries of the world, it continues to offer great prospects for economic growth and industrial development within the developing economies.

The spectacular growth of the software industry in some non-G7 economies has aroused both interest and concern [1]. For example, few areas of production, engineering or education do not include software as an important and increasingly complex component [32]. Moreover, new small firms with relatively few tangible assets can still prosper and grow rapidly and with the rise of interest, where these firms are physically based is becoming less important. However, because of the unique way that the knowledge is generated and traded in the software industry and other knowledge-intensive industries, protection of intellectual property forms a fundamental elements as to how the sector has grown and developed. It was discovered that the software industry has the potential to become one of the most internationally dispersed high-tech industries, concluding that in the last two decades, there have been high growth rates of this sector and a dramatic increase in the spread of computer software and services world-wide with a tremendous high level of productivity [21]. Therefore, governments, all over the world struggle with the problem of how to ensure that Science, Technology and Innovation (STI) contribute effectively to solving national problems [3].

In addition, software has become a key facilitating technology making it a major strategic technology for growth and development. Subsequently, software and computer services thus become centrally an underpinning not only the actual creation, but also the efficient utilization of core aspects of modern manufacturing and the physical products that are produced [2]. Indeed, the software industry y has become a leading source of employment creation and economic growth in the world [27].

II. Statement Of The Problem

Information technology is not the course of global challenges we are living through. But without ICT and software development as its components, none of what is changing our lives would be possible. The entire realm of human activity depends on the power of information that accelerates its pace. However, software

development is making possible user-friendly computing when users are provided with adequate education and can progress in that knowledge and ability to create wealth.

Technology per se does not solve social problems. But 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 stimulating development is a two-edge sword. On the other hand, it allows countries to leap frog stages of economic growth by being able to modernize their production systems and increase their competitiveness faster than in the past. On the other hand, it is the essential tool for economic development and material well-being in our age; its conditions power, knowledge and creativity; it is for the time being, unevenly distributed within countries; and it requires, for the full realization of its development value, and interrelated system of flexible organizations and information-oriented institution. In a nutshell, cultural and educational development conditions software development, which conditions social development and this stimulates cultural development and educational development. Hence, a virtuous circle of spiral development process which needs to be directed by the conflictive dynamics of the society and by technology.

Furthermore, econometric studies have shown close empirical relationship between diffusion of information technology, software development, productivity and competitiveness for countries, regions, industries and firms [16]. They also show that an adequate level of education in particular, is essential for the design and productive use of software development processes and products as a critical tool to drive new technologies and diffuses by the society [25]. In spite of the global evolution and revolution of software development as a key facilitating strategic tool for growth and development, such phenomenon seem not to have resonated sporadically, to the Nigeria environment and therefore software development has not to have keep pace with the dynamic nature of the sector and global trends.

The challenges of the Nigerian software industry are multivariate and it includes lack of development infrastructure for software, dearth of skilled manpower, knowledge incubation and professionals. Others are low patronages of indigenous software products and stiff competition by foreign software products. And more importantly, is the lack of continuity on indigenous software development efforts. This has resulted in most software development projects being abandoned haphazardly without the documentation of lessons learned. This lack of knowledge incubation on software development in Nigeria has stiffed rather than stimulate rapid development in the sector with attendant reflection on all other sectors of the economy.

III. Software Development Contingencies

The variety of software development process models that are available presents a challenge in determining which model is most effective for a particular project. The traditional waterfall model is most appropriate for large scale, customized systems [14], situations where there is little technological innovation and a tight schedule [24], and "precedent" systems in familiar domains and with stable requirements [19]. However, the waterfall model works poorly in situations including interactive end-user applications [12], unfamiliar systems [18], and complex systems for which the requirements are not well understood [10, 24]. Hullet [21] argues that any project which uses the waterfall approach risks inadequate functionality, schedule overruns, and even abandonment due to its "monolithic" nature (as opposed to incremental or iterative approaches)].

The other evidence supports the belief that certain development models are most effective only for a specific class of development situations. For example, exploratory data analysis has found that software projects facing very challenging conditions require greater "management power" that include risk control, collaboration, requirements definition, conflict resolution, resource availability, and project planning in order to be successful [17]. The "user involvement" according to the literature contains considerable empirical evidence that, under conditions of high complexity and uncertainty about user requirements, development approaches involving user participation such as prototyping methods are most strongly related to project success [24; 35, 25]. The position of literature on reuse and factory models stipulates that, while it is primarily conceptual or descriptive, it likewise suggests that the success of these approaches is contingent on the software development task [9; 15; 19]. The literature clearly supports the view that relationships between software development process choice and development effectiveness are contingent, and that the contingent or deciding factor has to do with characteristics of the specific software project.

IV. Literature Review

Software development was seen to be a creative art-form, and to a degree, experimental. Software development is fast changing in line with technological advancements, global changes in the business environment, changing customer demands and expectations and increasing market competition. It is becoming commonly accepted that software and Information Technology (IT) is no more special than any other business tool. Traditionally, software development was a process of identifying a specific set of requirements and coding to those software development lifecycle, to produce applications that would dictate the business processes [31].

Nigam observed that software has now become a core competency and general-purpose technology that is critical to the global competitiveness of most industries (all companies have the same hardware but they compete with software) and to the effective deployment of government services (beyond the basics of data processing) in every country, regardless of its level of economic development [26]. The fundamental change to development is a result of a number of factors, including the increasing reliance on technology, the need to adapt to changing marketplace demands, the issue of maintenance and integrating legacy systems along with a growing requirement for effective management of IT. These issues are the driving force for the extinction of monolithic software systems [36]. The notion of monolithic system is being replaced by the creation of modularized, loosely coupled parts and components that can be re-used and readily integrated across applications. This relatively new and growing requirement for componentization of software is impacting the nature of the global software development industry as it evolves into a manufacturing industry.

4.1Stages in software development (Software Development Lifecycle (SDLC))

There are several different approaches to software development. Some take a more structured, 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. Most methodologies share some combination of the following stages of software development: Market research, Gathering requirements for the proposed business solution, Analyzing 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 lifecycle, or SDLC [20].

The different approaches to software development may carry out these stages in different orders, or devote more or less time to different stages. The level of detail of the documentation produced at each stage of software development may also vary. These stages may also be carried out in turn (a "waterfall" based approach), or they may be repeated over various cycles or iterations (a more "extreme" approach). The more extreme approach usually involves less time spent on planning and documentation, and more time spent on coding and development of automated tests. More "extreme" approaches also promote continuous testing throughout the development lifecycle, as well as having a working (or bug-free) product at all times. More structured or "waterfall" based approaches attempt to assess the majority of risks and develop a detailed plan for the software before implementation (coding) begins, and avoid significant design changes and re-coding in later stages of the software development lifecycle [30].

V. A Theoretical Background on Software Development Approaches.

The theoretical framework for this study was based majorly on the concept of social development theory which focus on underlying activities and results; recognition of inherent creativity of individuals and societies and implied influence or oppositions of external factors. Interrelated aspects of processes and activities such as education, knowledge, skills and its application [10, 22; 9]. It is a process which involves inter firms processes as well as the relationships between firms and their environment [25; 23) leading to technological and social relevance processes and the adoption of its approaches in software development firms. However, the theory underscored the point that society develops its own strategies in pursuit of its own goals. No external forces and agency can develop a society.

5.1 Software Development Approaches

The methodology for every software development has more or less its own approach. There is a set of more general approaches, which are developed into several specific methodologies. These approaches [12] are Waterfall: linear framework type; Prototyping: iterative framework type; Incremental: combination of linear and iterative framework type; Spiral: combination of linear and iterative framework type and Rapid Application Development (RAD): Iterative Framework Type.

5.1.1Waterfall 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 requirements analysis, design, implementation, testing (validation), integration, and maintenance [18]. The basic principles of the waterfall model (Davies and Nielsen, 1998) are:

(i) Project is divided into sequential phases, with some overlap and splashback acceptable between phases.

(ii) Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.

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/sign off by the user and information technology management occurring at the end of most phases before beginning the next phase.

5.1.*2Prototyping*

Software prototyping is the framework of activities during software development process by creating prototypes, i.e., incomplete versions of the software program being developed. The basic principles of prototyping [11] are :

(i) It is not a stand-alone, complete development methodology, but rather an approach to 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 room for ease-of-change that may become necessary during the development process.

(iii) The user is involved throughout the process, which increases the likelihood of user acceptance of the final output/product for implementation.

(iv) A 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 possible in some cases to evolve from prototype to a real working system by scaling-up the configuration/specification.

(vi) A basic understanding of the fundamental of business problem is necessary to avoid solving the wrong problem.

5.1.3Incremental

Various methods are acceptable for combining linear and iterative systems development methodologies. This is with the primary objective of reducing the inherent project risk by breaking a project into smaller segments and providing room for flexibility to easily make desirable changes during the development process.

The basic principles of incremental development [11] 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 incremental, 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).

5.1.4Spiral

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) The 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.

(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"[12].

(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 [20].

(iv) It begins each cycle with an identification of stakeholders and their win conditions, and end each cycle with review and commitment [33].

5.1.4Rapid Application Development (RAD)

Rapid Application Development (RAD) is a software development methodology, which involves iterative development and the construction of prototypes. Rapid application development is a term originally used to describe a software development process introduced by James Martin in 1991. The basic principles of RAD [33] are:

(i) Key objective is for fast development and delivery of a high quality system at a relatively low investment cost.

(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.]) It aims to produce high quality systems quickly, primarily through the use of iterative Prototyping (at any stage of development), active user involvement, and computerized development tools. These tools may include Graphical User Interface (GUI) builders,

Computer Aided Software Engineering (CASE) tools, Database Management Systems (DBMS), fourth-generation programming languages, code generators, and object-oriented techniques.

(iv) The key emphasis is on fulfilling the business need, while technological or engineering excellence is of lesser importance.

(v) Project control involves prioritizing development and defining delivery deadlines or "timeboxes". If the project starts to slip, emphasis is on reducing requirements to fit the timebox, not in increasing the deadline.

(v) This generally includes Joint Application Development (JAD), where users are intensely involved in system design, either through consensus building in structured workshops, or through electronically facilitated interaction.

(vi) An active user involvement is imperative.

(vii) It iteratively produces production software, as opposed to a throwaway prototype.

(viii) It produces documentation necessary to facilitate future development and maintenance.

(ix) The standard systems analysis and design techniques can be fitted into this framework.

5.1.6Other software development approaches

Other methods of software development are:

(i) Object oriented development methodologies, such as Grady Booch's Object-oriented design (OOD), also known as object-oriented analysis and design (OOAD). The Booch model includes six diagrams: class, object, state transition, interaction, module, and process [26].

(ii) Top-down programming: evolved in the 1970s by IBM researcher Harlan Mills (and Niklaus Wirth) in developed structured programming.

(iii) Unified Process (UP) is an iterative software development methodology approach, based on UML. UP organizes the development of software into four phases, each consisting of one or more executable iterations of the software at that stage of development: Inception, Elaboration, Construction, and Guidelines. There are a number of tools and products available designed to facilitate UP implementation. One of the more popular versions of UP is the Rational Unified Process (RUP).

(iv) Agile Software Development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. The term was coined in the year 2001 when the Agile Manifesto was formulated.

(v) Integrated Software Development refers to a group of software development practices and deliverables that can be applied in a multitude (iterative, waterfall, spiral, agile) of software development environments, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.

VI. Techniques of software development

The foregoing software development techniques are composed of one of the software development models used in conjunction with one or more methodologies. The techniques of prototyping, cleanroom, and object-oriented are ways to implement the waterfall, incremental, and spiral models. These techniques may be mixed and matched on a single project. Also, portions of a technique may be used without using all aspects of that technique. This means that a project using the spiral model may combine prototyping with object- oriented analysis and design and also use cleanroom testing techniques.

6.1Strengths and Weaknesses of Waterfall, Incremental and Spiral Models.

The Table 1 summarizes the strengths and weaknesses of the waterfall, incremental, and Spiral models.

Table 1: Strengths and Weaknesses of Models

	Waterfall	Incremental	Spiral
STRENGTHS			
Allows for work force specialization	X	X	X
Orderliness appeals to management	X	X	X
Can be reported about	X	X	X
Facilitates allocation of resources	X	X	X
Early functionality	X	X	-
Does not require a complete set of requirements at the onset -	X	X	
Resources can be held constant	X	-	-
Control costs and risk through prototyping	-	X	-
WEAKNESSES			
Requires a complete set of requirements at the onset	X	-	-
Enforcement of non-implementation attitude hampers analyst/designer communications	-	-	X

Beginning with less defined general objectives may be uncomfortable for management	-	X	X
Requires clean interfaces between modules	-	X	-
Incompatibility with a formal review and audit procedure	-	X	X
Tendency for difficult problems to be pushed to the future so that the initial promise of the first increment is not met by subsequent products	-	X	X

Source: Boehm, 1998 X – indicates type of strength and weaknesses possessed by each approach

VII. Research Methodology

The study covered some selected software development firms, users and educational institutions in the six geopolitical zones of Nigeria. The seventy-five small, medium and large ICT firms (Lagos 70; Abuja 2; Port-Harcourt 2 and Kaduna 1) involved in software development and other ancillary firms published in the Goldstar directories of 2007/2008 were purposively selected for the study. Data were collected using sets of questionnaire and from secondary sources. The questionnaire was administered on software developers in select ICT firms to elicit information on the various approaches adopted for software development. Another set of questionnaire was administered on educational institutions who are both developers and users comprising Heads of Computer Departments of Polytechnics (5) and Universities (6) purposively sampled from the selected tertiary institutions involved with software development in Nigeria. Two hundred copies of the questionnaire were administered out of which 183 (92%) were retrieved and 144 (79%) found useful and analysed. Secondary data was collected from publications on the operations and competitiveness of software development in Nigeria. The Descriptive statistics used was Duncan Multiple Range (DMR) to confirm one relative strength (weighted Mean) of each variable in the factor and inferential statistics using Analysis of Variance (ANOVA) to know the relationship among the variables were also employed for data analysis. Reliability and validity test were carried out and the results obtained showed that all the items had an alpha above standard guideline of 0.70. This implies that the scales are suitable for analysis with acceptable level of reliability. Cronbach's alpha score of 0.746 was obtained for the entire scale. This indicated internal consistency and reliability with satisfactory construct validity of the variables [28].

VIII. Results and Discussion

8.1Approaches for Software Development adopted by firms

Table 2 shows the descriptive analysis of some of the approaches adopted. These include: User centre (15.6%), waterfall (22.2%), prototyping (11.1%), computer/incremental case (6.7%), Integrated approach (13.3%), while rapid, spiral and agile approach had (4.4%) each. The weighted mean and standard deviation of these approaches were as well shown. User centre had a weighted mean of 1.64 and a standard deviation (SD) of 0.48; waterfall had a weighted mean of 1.83 and SD of 0.83; prototyping had a mean value of 2.16 and SD of 0.95; computer case had a mean value of 1.43 and standard deviation of 0.73; integrated approach had a mean value of 2.90 and SD of 0.92; rapid approaches development had a mean value of 1.79 and SD of 0.69; object oriented approach had a mean value of 2.18 and SD of 1.21; spiral approach had a mean value of 2.17 and SD of 1.18; agile approach had a mean value of 2.05 and SD of 1.19 while rotational unified process had a mean value of 2.28 and SD of 1.15. The result indicates that integrated approach had the highest weighted average (2.90) and a standard deviation of 0.92; which is comparatively lower to other close mean values and deviations.

The rotational unified process approach shows an average value of 2.28 and deviation of 1.15; Object oriented approach has an approximate mean value of 2.18 and standard deviation with sigma (б) of 1.21. This implies that integrated approach followed by prototyping, rotational unified process, spiral, agile and object oriented development approaches were moderately employed approaches in software development in Nigeria. The values for User Centre, Water fall, Computer/incremental case and Rapid approaches however lower respectively. This suggesting that development of software based on these approaches is low and seldom adopted by software firms in Nigeria. This also implies that integrated approach followed by prototyping, rotational unified process, spiral, agile and object oriented development approaches were moderately employed approaches in software development in Nigeria. The value for user centre, water fall, computer/incremental case and rapid approach development are however lower respectively. This shows that development of software based on these approaches is low and seldom adopted by software firms in Nigeria.

Table 2: Approaches Adopted for Software Development

Variable	Frequency	Percentage (%)	Weighted Mean	Standard deviation
User centre	7	15.6	1.64	0.48
Waterfall	10	22.2	1.83	0.38
Prototyping	5	11.1	2.16	0.95
Computer/incremental case	3	6.7	1.43	0.73
Integrated	6	13.3	2.90	0.92
Rapid approaches development	2	4.4	1.79	0.69
Object oriented development	2	4.4	2.18	1.21
Spiral	3	6.7	2.17	1.18
Agile	5	11.1	2.05	1.19
Rotational unified process	2	4.4	2.28	1.15
Total	45	100		

8.2Correlation Matrix of the Software Approaches

It can be observed (Appendix 1) that most variables with the exception of Rapid and User Centre approach display considerable variation. The correlation coefficient between waterfall and user centre approach is positive (0.25) showing a direct relationship between the two variables. A reverse trend is however observed between water fall and prototyping (-0.23). Similar negative trends were observed between prototyping and user centre (-0.28), prototyping and integrated (-0.13), prototyping and rapid (-0.03) as well as integrated and computer (-0.01) - indicating inverse relationships between variables showing negative signs.

The correlation coefficients among the remaining variables are all positive and considerable. However, the positive correlation coefficient observed between rapid approach and user centre approach (0.87) is high suggesting a possibility of collinearity between the two variables. This indicates a need for normality test for all the variables.

8.3An In-Depth Evaluation of the Approaches for Software Development in Nigeria

After several diagnostic tests on the variables specifying approaches for software development, an in-depth evaluation of each of the approaches was analyzed using confirmatory factor analysis. The result of the confirmatory factor analysis is presented in Tables 2, 3, 4, 5 and 6. Table 2 shows total variance accounted for by each factor. The Eigen values associated with each factor represent the variance explained by that particular linear component (Scree plot of eigen values Figure 1). The first few factors explain relatively large amounts of variance whereas subsequent factors explain only small amounts of variance. Kaiser criterion Bandalos and Boehm-Kanfman [7] suggests to retain those factors with eigenvalues equal or higher than 1.

Based on this, the first two factors User centre (4.27) and Waterfall (1.79) are relevant for this study thus justifying their retention in the subsequent analysis and discussion. The rest of the factors 3 to 10 all have Eigen value below 1 and therefore fall short of consideration based on Kaiser Criterion. The third column in Table 3 indicates the

Table 3: Total Variance Factor of the Model (Component matrix)

Factor	Eigen value	Difference	proportion	Cumulative
User Centre	4.27	2.49	0.43	0.43
Waterfall	1.79	0.82	0.18	0.61
Proto-typing	0.96	0.15	0.10	0.71
Computer/ Incremental	0.82	0.02	0.08	0.78
Integrated	0.79	0.32	0.08	0.86
Rapid dev.	0.47	0.14	0.05	0.91
Object oriented	0.33	0.07	0.03	0.94
Spiral	0.26	0.03	0.03	0.97
Agile	0.23	0.16	0.02	0.99
Rotational Unified process	0.07	-	0.01	1.00

chi2(45) = 1699.06 Prob>chi2 = 0.00

proportion of difference between one Eigen value and the next. Since the sum of Eigen values equals total number of variables, proportion indicates the relative weight of each factor in the total variance. Thus, the first factor explains 42.73 per cent of the total variance. The cumulative shows the amount of variance explained by n + (n-1) factors. For example, factor 1 and factor 2 in Table 5.3 account for 60.60 per cent of the total variance. This, by inference, affirmed User centre and Waterfall with over 60 percent relative weight vide Table 3 are the

most commonly used approaches by Nigerian firms for software development. The Screen plot showing the pictorial representation of the components is shown in Figure 1. The plot of Eigen value was suggestive of 2 factor loading.

8.4 The Dimensional Uses of the Approaches

The dimensional (extent of) uses of the various approaches were subjected to factor analysis test. Factor loadings are the weights and correlations between each model of software development approaches. The higher the load, the more relevant is the variable in defining the factor's dimensionality. A negative value indicates an inverse impact on the factor. Here, two factors are retained because both have eigenvalues over 1. Table 3 shows that computer/incremental, rapid, object oriented, spiral, agile and rotational unified process approaches seem to define factor1, while user centre, waterfall, integrated and rapid approaches define factor 2. Similarly, the uniqueness of the factors was also tested. Table 4 is a reconfirmation of validity of factors of variance.

Uniqueness is the variance that is 'unique' to the variable and not shared with other variables. It is equal to 1 –communality (variance that is shared with other variables). For example in Table 3, 26% of the variance in 'user centre' is not shared with other variables in the overall factor model. On the contrary 'Agile' has low variance not

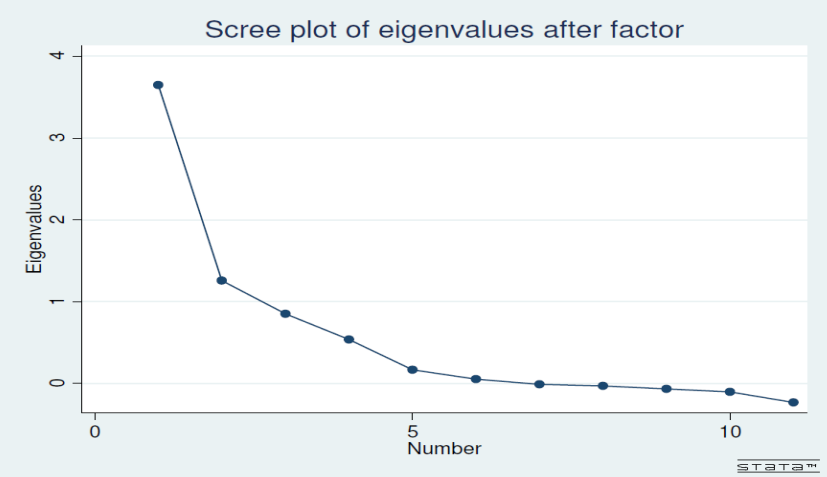

Fig.1: Graphical representation of the approaches of software development

Accounted by other variables (24%). The greater the value of 'uniqueness' is, the lower the relevance of the variable in the factor model. Thus, integrated approach, waterfall approach and computer/Incremental approach have low relevance in our factor model.

8.5Matrix Rotation of the Variables

Table 4 shows the results of the two factors using an orthogonal rotation. This is a matrix of the factor loadings for each variable on to each factor. This matrix contains the same information as the component matrix except that it is calculated after rotation (Table 5). By default the rotation is varimax which produces orthogonal factors. This means that factors are not correlated to each other. This setting is recommended in order to identify variables to create indexes or new variables without inter-correlated components. As in previous component factors (Table 3) this result indicates that both factors explain 60.60% of the total variance observed.

This result underscores the position in the literature on software capacity as being directly correlated with the size, knowledge and skill of the available workforce [34]. While workforce mobility is also being cited as a major advantage of Silicon Valley since the practice shorten the learning curves across all companies [32].

The pattern matrix in Table 5 offers a clearer picture of the relevance of each variable in the factor. Factor1 (with 0.39 cumulative) is mostly defined vide table 5.3 by computer/incremental case (0.61), object oriented development (0.85), spiral (0.82), agile (0.83) and rotational unified process (0. 82) while factor 2 (with 0.61 cumulative) is defined by user centre (0.60), rapid approaches development (0.78), integrated (0.27) and waterfall (0.30) approaches.

Tables 5 and 6 shows the regression coefficients used to estimate the individual scores (per case/row). The result shows that the coefficients of Water fall, Integrated approach negatively impacted in factor 1 while Prototyping, Computer, Object, Spiral agile and rotational approach negatively impacted in factor 2. Overall, the

confirmatory factor analysis indicates that user centre, object oriented, spiral, agile, and rotational unified process approaches are the most relevant approach for software development in Nigeria.

Table 4: Un-rotated Pattern Matrix and Unique Variances

Variable	**Factor1**	**Factor2**	**Uniqueness**
User center	0.60	0.62	0.26
Waterfall	0.30	0.49	0.67
Prototyping	0.15	-0.74	0.44
Computer/incremental case	0.61	-0.27	0.56
Integrated	0.27	0.50	0.68
Rapid approaches development	0.78	0.37	0.26
Object oriented development	0.85	-0.18	0.25
Spiral	0.82	-0.20	0.29
Agile	0.83	-0.25	0.24
Rotational unified process	0.82	-0.18	0.30

Table 5: Rotation: Orthogonal Varimax (Kaiser off)

Factor	Variance	Difference	Proportion	Cumulative
Factor1	3.93	1.79	0.39	0.39
Factor2	2.14	-	0.21	0.61
chi2(45) = 1699.06				
Prob>chi2 = 0.00				

Table 6: Rotated factor loadings (pattern matrix) and unique variances

Variable	Factor1	Factor2	Uniqueness
User centre	0.32	0.80	0.26
Waterfall	0.10	0.56	0.67
Prototyping	0.41	-0.63	0.44
Computer/incremental case	0.66	-0.02	0.56
Integrated	0.06	0.57	0.68
Rapid approaches development	0.59	0.63	0.26
Object oriented development	0.85	0.15	0.25
Spiral	0.84	0.12	0.29
Agile	0.87	0.08	0.24
Rotational unified process	0.82	0.14	0.30

Table 7: Scoring coefficients (method = regression; based on varimax rotated factors)

Variable	Factor1	Factor2
User centre	0.00	0.38
waterfall	-0.04	0.28
prototyping	0.19	-0.37
computer	0.19	-0.09
integrated	-0.05	0.28
rapid	0.09	0.26
object-oriented development	0.22	-0.02
spiral	0.22	-0.03
agile	0.23	-0.06
rotational unified process	0.22	-0.02

IX. Conclusion

From the study, it can be concluded that software development firms in Nigeria commonly adopt integral approach, prototyping, rotational unified process, agent-oriented, spiral and agile approaches for their software development operations, while object oriented development approach were moderately employed. The extensive use of these approaches was informed by the optimal nature of the approaches; they (that is, the respondents) emphasize on completion of a phase of the development before proceeding to the next phase; and also the fact that they provide basic understanding of the fundamental business problems and thus avoid solving the wrong problems.

X. Contributions of this Research work to Knowledge

Scholars have made efforts to explain the dynamic nature of software development as a diversified concentrates on examining different activity with many interacting components. A conceptual framework that examines these relationships and their various context was devised for the study. Thus, this study concentrates on examining different approaches used in software development to determine the users at large. The study also

revealed that either internal (skills, education experience etc) or external (market share, competition etc) factors may determine and/or influence the software development approaches adopted in the firms.

These factors interact to determine the functional strength available in the firms relative to investment and production capability. The performance and mastery of these functions generate the capabilities of various approaches that software developer adapts and improves upon for the processes that generate the products in the industry.

Acknowledgement

We remain grateful to all that contributed to the success of this research work. God is the source of life, wisdom, knowledge and everything that is good. Finally, we acknowledge with gratitude the invaluable support received all through the duration of the research.

References

[1]. A.. Abran, L. LaFramboise, and P. Bourque, "A Risk Assessment Method and Grid for Software Measurement Programs", Software Engineering Management Research Laboratory, 3 October, 2000. Http://www.Irgl.uqam.ca/publications/pdf/271.pdf Retrieved on November, 2009.
[2]. J. A. Alic, "Technology in the service industries", International Journal of Technology Management 2005, 9 (1), pp. 1-14.
[3]. A.O. E . Animalu, 'Technology Policy and Funding in Nigeria', Science and Technology News: a quarterly Publication of the Federal Ministry of Science and Technology, Abuja. July-Sept. 2002, 40 – 45.
[4]. M. K. Badawy, Technology Management Education: alternative Models California Management Review, 2008, 40 (4), pp. 94 – 115.
[5]. S. Bagchi, "India's software industry: the people dimension", IEEE Software May/June. 1999, pp. 62-65.
[6]. T. Bandalos, and P. Boehm-Kanfman, "Training and Productivity in African Manufacturing Enterprises." Paper presented at the Conference on Enterprise Training Strategies and Productivity. Private Sector Development Department,World Bank,(Washinton, DC, 2009).
[7]. A. Bedi, "The role of information and communication technologies in economic development: a partial survey",(ZEF-Discussion Papers on Development Policy No. 7,Center for Development Research, ZEF, Bonn 19999).
[8]. D. Bell, Constructing Social Theory. Lanham, MD: Rowman & Littlefield. ISBN 978-0742564282 Sociology and History of Technology. Cambridge, MA: MIT Press, 2008.
[9]. R. Bijker, E. Wiebe., P. Thomas, W. Hughes, and J. Trevor, The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology. Cambridge, MA: MIT Press, 1997.
[10]. J. D. Blackburn, G. D. Scudder, and L. N. Van Wassenhove, "Improving Speed and Productivity of Software Development : A Global Survey of Software Developers." IEEE Transactions on Software Engineering. 1998, 22(12), pp . 875-885.
[11]. B.W. Boehm, Software Risk Management,(1998, Washington, D.C.: IEEE Computer Society Press).
[12]. J. Bowen, and V. Stavridou, , "Safety-critical systems, formal methods and standards"Software Engineering Journal, 2007,8 (4), pp. 189-209.
[13]. T. E. Clarke, Why do we still not apply what we know about Managing R&D Personnel? Research Technology Management. March – April 2002 ,45, (2), 9 – 11.
[14]. M . A . Cusumano, Japan's Software Factories: A Challenge to U.S.Management. (New York : Oxford University Press, 2001) .
[15]. L. J. Davies and S. Nielsen, An Ethnographic Study of Configuration Management and Documentation Practices in an Information Technology Centre. (1998, Elsevier/North Holland, Amsterdam).
[16]. M. S. Deutsch, and R. R. Willis, , Software Quality Engineering, (2008, Englewood Cliffs, NJ: Prentice Hall).
[17]. F. Gaio, .and K. Brazilian Guha, "Software strategies for developing countries: lessons from the international and "Indian groups face struggle to protect their software arms", Financial Times, 28 August 1998.
[18]. K. Guha, "Indian groups face struggle to protect their software arms", Financial Times, 28 August 1998.
[19]. R. Heeks, India's Software Industry: State Policy, Liberalisation and Industrial Development (New Delhi and London: Sage Publications, 2005).
[20]. D.T . Hullet, "Key Characteristics of a Mature Risk Management Process", Fourth European Project Management Conference, PMI Europe, 2001.
[21]. G. Jacobs, and H. Cleverland, Social Development Theory. International Centre for peace and development, 1999. 2352 Stonehouse Drive, Napa, CA94558, USA.
[22]. B. Lundavall, and B. Johnson, "The Learning Economy". Journal of Industry Studies, Vol. 2, No. 1, 2004
[23]. J. A. Mahmood, , "High technology industrialisation in East Asia". Journal of Industry Studies, 3, 1999, pp. 1-77.
[24]. K. Mckeen, L .Van Wassenhove, and S. Dutta, "Performance Evaluation of General and Company-Specific Models in Software Development Effort Estimation "Management Science. 2009, 45(6), pp. 787-803.
[25]. J. K. Nigam, Structuring of Research Institutes to promote Integration with Industry in Mengu, M. d. (Ed.). R & D Institute and Local Industry Interaction: An International perceptive. WAITRO, 2000, 256 – 271.
[26]. OECD, Science, Technology and Industry Scoreboard 2007, Innovation within Companies, 2007a, pp 34.
[27]. D. L. Pallant, Four common misconceptions in exploratory factor analysis in Statistical and methodological myths and urban legends: Doctrine, verity and fable in the organizational and social sciences. Lance, Charles E. (Ed.); Vandenberg, Robert J. (Ed.). (New York: Routledge, 2004) pp. 61–87.
[28]. R. S. Pressman, A Manager's Guide to Software Engineering, (1999 New York: MsGraw- Hill).
[29]. V. Prochnik, and P. Quéau, "The Brazilian software production and export industrial policy: Softex-2000", Economics Institute, Federal University of Rio de Janeiro, mimeo. "Who owns knowledge?" Le Monde Diplomatique (English Edition), January 2000.
[30]. J. J Rakos, Software Project Management for Small to Medium Sized Projects, (Englewood Cliffs, NJ: Prentice Hall, 1999).
[31]. R. Schware, "Software Industry Entry Strategies for Developing Countries: A "Walking on Two Legs" Proposition", World Development, 2000, 20(2). Pergamon Press, pp. 143 – 164.
[32]. M . V. Tatikonda, and S . R. Rosenthal, "Successful Execution of Product Development Projects Balancing Firmness and Flexibility in the Innovation Process ." Journal of Operations Management. 2000, 18(4), pp. 401-426.
[33]. L. T. Tesler, and C. K. Barr, "Information technology and the Internet: the Singapore experience", Information Technology for Development, 2009, 8.

[34]. C. Thor**,** Designing Feedback: Performance Measures for Continuous Improvement. Crisp Publications, Menlo Park, CA, 1998, p.3.
[35]. N. Whitten, Managing Software Development Projects, Second Edition, (New York: John Wiley & Sons, Inc, 2005).

Appendix 1

Correlation matrix of the approaches for software development

Variable	User	Waterfall	Proto-typing	Computer	Integrated	Rapid	Object	Spiral	Agile	Rotational
User centre	1.00									
Waterfall	0.25	1.00								
Proto-typing	-0.28	-0.23	1.00							
Computer	0.22	0.11	0.30	1.00						
Integrated	0.30	0.26	-0.13	-0.01	1.00					
Rapid	0.87	0.25	-0.03	0.48	0.25	1.00				
Object	0.34	0.17	0.19	0.45	0.20	0.49	1.00			
Spiral	0.30	0.18	0.15	0.42	0.11	0.47	0.72	1.00		
Agile	0.28	0.17	0.20	0.44	0.10	0.47	0.74	0.76	1.00	
Rotational	0.34	0.13	0.18	0.39	0.14	0.51	0.70	0.66	0.71	1.0000