International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Index Copernicus Value (2016): 79.57 | Impact Factor (2017): 7.296

Web Application Development Issues and e-business Software Development Life Cycle

Patrick N. Kiratu[1], Felix N. Musau[2]

[1]PhD Student, Kenyatta University, School of Business, City Campus, P.O. Box 43844-00100 Nairobi, Kenya

[2]School of Computer Science and Sciences, Riara University, Mbagathi Way, P.O. Box 49940-00100 Nairobi, Kenya

Abstract: ***E-transaction applications have become indispensable in today's competitive business environment, and are widely used in several areas such as commerce, education and manufacturing among others. They enhance ubiquitous access of goods and services in a short time and fast. Developing high quality e-business applications is the main goal software engineers who make use of a variety of clearly defined development models in solving clients' demands. The choice of a software development model is dependent on its anatomy, challenges it poses and response to emerging technological advancements. The present study reports on an appraisal of software development approaches used in practice. The study investigates the issues faced in using the models, the various models structure and their suitability in coping with emerging technological trends. The study shows a discrepancy in existing software developing approaches appropriateness in adequately addressing the prevailing e-business environment specifications. A new software development approach is proposed, in this context, which is applicable to many e-business projects.***

Keywords: e-business, challenges, models, waterfall, Interactive

1. Introduction

The web has been recognized as a dominant medium of new and increasingly changing information in modern times. Today, more and more businesses continue to set up websites to market and sell their goods or services. Contemporary customer is more informed and more demanding than ever before. The advent and extensive adoption of web 2.0 platform dictates that the consumer be more involved in virtually all the processes business, from production to delivery and even maintenance of the product or service. The client makes contributions on how a good or service is to be provided, when it is to be supplied, the mode of delivery and provides feedback on its quality. This poses challenges not only to businesses but also to software developers. The businesses face a risk of being rendered irrelevant in case they ignore the consumer participation; although, the extent to which the client is to be involved in the business processes is not resolute. On the other hand, software developers face a challenge of incorporating most of the ever-changing consumer requirements in application designs, while ensuring that they do not compromise on security of information which is critical in e-business transactions. This study examines the existing software development life cycle models for e-business application development and proposes an integrative waterfall model that caters for consumers' requirements in all the phases in concurrence with emerging technological trends such as web 2.0 platform.

1.1 Web Application Development

A web application is an end-user program, stored on a remote server and accessed over the Internet by a browser [12]. Since 1994 when Internet became available to the public, it has continued to host many, increasingly sophisticated and innovative websites. The quality of a commercial websites can be viewed from two perspectives: static and dynamic. The static quality of a website is the analysis of the in terms of design of the elements in relation to the purpose, content and structure of the site while dynamic aspect is related to customer interactivity and feedback [25]. According to [25], the latter approach achieves a measure of social acceptability by ensuring that content and products presented are in harmony with the living habits, culture and social system of the target population. Growing access of the Internet through mobile devices with different specifications and capabilities have brought on board an increased number of users: skilled and unskilled, with varied interests and goals thereby broadening the dynamic requirement for websites.

An ubiquitous web modeling approach responsive to mobile devices was proposed by [42] who concurred that the existing web modeling languages lacked proper engineering design foundations which hampered their scope and capabilities in a dynamic environment. The advent of Web 2.0 social network platform bridged the interactivity gap identified in the preceding approaches. In this paradigm peers contribute to the development tools, content and among communities on the Internet [43]. The approach embraces the use of highly interactive social media platforms to harness knowledge that boosts user experience, bringing forth a unique and distinct source of information that is disruptive to the legacy markets. Web 2.0 service models, such as social networking sites, video-sharing sites, wikis, blogs and web-based communities, allow for information to flow in both directions and are not constrained by time or location. These participative technologies emphasize on exchanging consumer-generated content, easy to use applications and interoperability between different end-users computing devices [4]. To tap into the business opportunities inherent in these platforms, businesses continue to engage stakeholders in the design, deployment and access to their websites. This is a trade-off of business opportunity versus challenges posed by these technological advances.

2. e-Business Application Development Issues

Nowadays firms leverage on web technology to maintain competitive advantage by deploying more innovative and dynamic applications. This is accelerated by the customers demand to know the different aspects of products or services on offer such as cost, look, feel and ultimate value for money. Clients often achieve this by visiting businesses websites or social media platforms using computers or mobile computing devices. In developing a successful e-business presence, factors such as website responsiveness and user-friendliness are critical [39]. However, several issues hinder this achievement and can be categorized into two broad groups. There are those e-business software issues that the client demands like ease of use and inter-operability with different computing devices and the challenges firms faces in satisfying the customers' needs, like security of data [26]. The two categories are interdependent and are addressed almost simultaneously. The former are more challenging due their diverse nature and volatility and have an impact on the latter.

2.1 Technological Issues

Nowadays, many businesses are inclining towards e-transactions to enhance competitiveness in their industries. On the contrary, [26] observed that introducing new e-business applications to businesses that were not accustomed to using such technology results in challenges and the firm struggled to operate under the new standards. The issues were aggravated by migrating from static to dynamic websites causing many firms to struggle optimizing the opportunities provided by the newer technology leading to plummeting of profits. Firms also, face challenges on how to select the appropriate method of web application development based on critical factors of quality, time and cost. The market offers a wide variety of e-business application developers and tools whose choice and success depends on a number of factors, some known to the firm and others unplanned for. In addition, today's business applications need to be more flexible, scalable and interoperable to cope with a diverse and rapidly changing technological and e-commerce environment.

2.2 Strategic Issue

Dynamic web applications and web 2.0 platforms are interactive in nature and have opened up businesses to customers [26]. This has caused slacker firms indecision on how much operational transparency to provide to users during e-transactions and how much business control to relinquish. [26] adds that these businesses face challenges of determining the extent of investment necessary to comply with new standards and how to formulate business models that can be understood by consumers. Other strategic issues identified, that obscure the already existing threats borne by firms in an industry, include: extent of information sharing dilemma, abandoning proprietary legacy applications, changing the business cultures and the extent of product ownership to surrender.

2.3 Responsiveness of Applications

According to [26], earlier e-business applications were static, rigid, and unresponsive forcing users to resist them due to poor design. This was followed by websites that focused on users, but often restricted themselves to a fixed set of perspectives. Again, users resisted them due to lack of skills and usability errors. Owing to the dynamic, complex and unpredictable nature of users, e-business applications designs have accommodative interfaces and workflows that match pre-defined usage contexts – like ability to continue designing while still in use [39]. Hence, [26] affirms that designing of such responsive e-business applications is tedious, expensive and difficult for many businesses in an industry.

2.4 Proficiency Issues

Designing web applications to support several end-user devices while simultaneously, offering personalized services require adequately software engineers. There are several and varied e-business application developing tools in the market leading to a multitude of solutions that can be deployed and maintained. Consequently, web hosting companies get strained by the increased demands and often lack staff with requisite skills to handle the mutating designs [8]. At the business level, specialists are required to address e-transaction issues that may arise from time-to-time. Businesses that are unable to resolve end-user issues originating from e-commerce are likely to incur losses when transactions are abandoned or dissatisfied customer fade away. In addition, website down-times are not pleasant for businesses in a competitive environment leading higher demand for e-business application specialists.

2.5 Complexity Issue

In designing e-transaction applications, content is created dynamically based on prevailing status of businesses, such as stock levels and prices, hosted in database systems. Web servers are deployed and integrated to link the business systems at the backend to the Internet. These arrangements coupled with the need for continuous update of user interfaces have a bearing on transactions response time which is significant in virtual environments [24]. The conglomeration of devices bearing diverse applications increases intricacies to the already existing network components making it difficult to troubleshoot and fix in case of downtime.

2.6 Security

Embracing Web 2.0 platform by many firms and users have bred a new generation of Internet users in the workplace bringing with them the comfort of social media. [40] observes that the net-generation has embraced technology in both private and professional live and has a different perspective of organization work culture, access to information and multi-tasking. This has brought a conflict between members of the generation and the rigid policies prevalent in many businesses [40]. Social platforms have developed fast forcing businesses to shift their focus to providing users access to data and resources without considering the risks involved. Research has identified security threats posed by allowing users access to firms'

networks as electronic intrusion by hackers or malicious software attacks. The business may also experience data leakage, loss of confidentiality and privacy which often result in brand damage, tarnishing of organization's reputation or loss of intellectual property rights [39].

2.7 Legal and Ethical Issues

[39] claims that allowing customers to a businesses network, as is the case in web 2.0 platform, exposes it to both legal and financial penalties from regulatory authorities arising from compliance breaches, copyright infringement or plagiarism since shared information may be factually wrong, untrustworthy and difficult to authenticate the sources. Additionally, combining information from various sources could diminish its relevance leading to suboptimal use of organizational resources and time. The businesses are also, likely to suffer losses arising from abandoned operations or inadequately maintained or tested applications occasioned by legal caveats.

The issues identified in e-business application development processes point to the significance of choosing and adopting models that mitigates them while maintaining user-acceptability and developers' satisfaction. This study investigates the structure and suitability of various software development approaches in the context of changing consumer requirements. The study categorizes the approaches into conventional/traditional approaches, contemporary models, emerging approaches and a proposed interactive waterfall model.

3. Software Development Life Cycle

A cycle is a series of occurrences repeated routinely within a given duration where events repeat themselves [35]. Therefore, a life cycle is a succession of events or blueprints that reveal themselves in the existence of an entity [30]. Software development life cycle is central to successful completion of an e-business application. The determination of whether the life cycle is formally implemented or superficial lies with the clients and the application developers.

3.1 Classical Software Development Approaches

These were the initial software development models and have been in existence for some time. They have been used repeatedly in small and large projects of varying complexity. This study investigates the anatomy and appropriateness of four contemporary software development life cycle approaches in addressing e-business application development life cycle challenges. They include: the code and fix model; the waterfall model; evolutionary model and the spiral model.

3.1.1 Code and fix Model

Between 1950s and 1960s, software development was a single person task dictated by the fact that: it was a science, the developer was also the user, requirements were known and the development of the application involved coding and fixing bugs that existed [30]. However, this model proved inadequate as computers became popular and developers distinct from users. This motivated software engineers' aspiration to incorporate user requirements when designing applications thereby making them more complex, necessitating collaborative approaches. Consequently, software products began to fail because identifying flaws and correcting them was difficult. Important issues of quality assurance testing; changing user requirements and documentation paved way for newer software and more systematic way of developing the applications [30]. Code and fix approach is suitable for small projects where the user is the same one using it and efforts directly contribute the product. However, the resulting model rarely matches the user needs and is prone to errors that are costly to fix and several reworks leads to code deterioration [44]

3.1.2 The Waterfall Model

The waterfall model derives its name from its geometric shape similarity with a typical waterfall. There are several varieties of the model phases emanating from different amount of details given by varied scholars and the manner of categorization [9]. The model consists of several non-overlapping phases: requirements analysis and specification, detailed design, coding, testing and implementation/ maintenance [37]. In waterfall model, the project is sub-divided into sequential stages with some overlie and allow loop back between the phases. It stresses on planning, time schedules with target dates, budgets and implementation of the whole system at once. Finally, control of software project is enhanced through extensive documentation, official reviews and endorsement by the end-user and developers upon completion of a phase.

According to [3], the model is used to implement major software projects in government agencies and large companies. It is easy to use, even with inexperienced teams and reduces planning costs. However, [30] notes that it has some shortcomings in form of budget overruns, late or suspended deliveries and overall dissatisfied clients. In addition, its emphasis on perfect analysis and design often result in too many meetings and too much documentation, substantially delaying the process of integration and testing. This leads to late delivery of projects characterized by poor quality, costly rework or unfeasible applications. [5], observes that during the requirements elicitation phase the risk of omitting crucial elements is high and unpredictable. Although the risks get stabilized through the phases, their late resolution results in late design changes and code with low fixes. [30] concedes that the approach characteristic requiring specification of all user requirements comprehensively and unambiguously at the beginning assumes that all of them are significant and remain change throughout the process which is unrealistic.

Although waterfall model provides the much-needed guidelines for a disciplined approach to software development, [3] observes that it is rigid since the results of one phase are to be frozen before the next phase could begin. It is monolithic since all planning is geared towards a single delivery date and heavy document-driven to the point of being bureaucratic. In addition, real application developments rarely followed the chronological flow depicted by the model and a slight alteration causes confusion as the project progresses [30].

3.1.3 The Evolutionary Model

The evolutionary model is grounded on continuous advancement of application development process in response to changing user requirements. It is a refinement of the waterfall model aspect where the consumer does not get to know anything about the software until the end of the project [13]. The model involves customers by developing and presenting working models of the software for their feedback. [30] categorizes evolutionary approach into two: incremental and prototyping approaches. In incremental approach, software product is developed in stages with some sections of the application postponed and additions occurring only to enhance its functional capabilities. Gradually, other functions are incorporated as enhancements, a move viewed as a replica of the waterfall process model where coding, integration and testing are done in an incremental style [13]. On the other hand, prototyping software development process is based on an investigational procedure typified by developing a working model of the application and giving it to end-users for comments and feedback. It takes two forms: throwaway prototyping – where initial version of the software is developed only temporarily to elicit user requirement information then discarded; or evolutionary prototyping where the initial design is progressively transformed into a workable application [44].

The model is suitable for both small and medium projects. It is beneficial in that feedback generated from earlier increments is used to enhance later stages by enabling users to understand their needs early and also takes a shorter time to development an application. However, the software deteriorates due to late increments that may require adjustments to earlier stages. In addition, it requires skilled personnel and expert management [44]. Again, not all projects are divisible into functional units and programmers are more productive working on large systems as opposed to small modules fronted by the model [1].

3.1.4 The Spiral Model

Spiral model explains the importance of iterative development by incorporating the characteristics of the waterfall model and the prototyping approaches [3]. It depicts application development process in form of a spiral that curls in a clockwise manner where iteration represents a phase of the waterfall model. Each phase commences with a design goal and terminates with the consumer undertaking an appraisal of the progress made. The model combines analysis and design as the applications get completed [37]. There is a deliverable for each cycle of the spiral in the model, although, it presupposes no fixed phases and grants this prerogative to the client - leading to numerous variations in the number of iterations, from business-to-business, assignment-to-assignment and customer-to-customer. Each quadrant of the spiral corresponds to a particular set of activities for all phases. They include: determining objectives, evaluating alternatives, developing the next level product and finally, planning next phases. The radius of the spiral signifies the cumulative cost of development while the angular dimension represents the progress [30].

According to [3] the model is advantageous in delivering projects early in the life cycle and takes strengths of other models. Additionally, its risk-driven approach avoids potential reviews and rework leading to higher customer approval rates. Moreover, the approach is more flexible since additional features can be added to the system in future. However, the approach may be expensive, escalates project duration, requires expert management in risk analysis and also, generates lengthy documentation - hence undesirable for small projects [30].

3.1.5 Other Classical Models

The number of traditional software development models was compounded by [10], who proposed two additional approaches. The reusable software – where past workable designs and code are recycled in new projects; and the automated software synthesis – where user specifications or designs are automatically converted into workable projects using high-level programming languages with code generators or CASE tools. However, these approaches may be viewed as subsets of the mainstream models in the classical or contemporary categories. For instance, software reuse is considered as a form of component based software engineering, while the automated software synthesis as a subset of extreme programming or prototyping.

It is evident that most conventional software development life cycle models diminished flexibility, enforced misuse of resources, took long to deliver systems and lacked proper documentation among other shortcomings. The models never addressed the twin issues of cost and quality together and often promised one at the expense of the other. This lead to emergence of modern methods of e-business application development approaches.

3.2 Contemporary Software Development Approaches

Research has shown that the classical software development approaches faced a number of challenges ranging from exceeding budget, long delivery period and complexity. Many projects failed to achieve the set targets and others had to be reworked leading to consumer dissatisfaction. Several alternative approaches, which were iterative and incremental in nature, emerged to fill the gaps identified. They aimed at addressing the shifting user requirements and advancement in technology. This study investigates the structure and suitability of the component based software development model, rational unified process, win-win spiral model, rapid application development (RAD), cleanroom engineering, concurrent engineering and agile development process.

3.2.1 Component-based Software Development

It is based on object-oriented programming that uses classes that encapsulate both data and methods [18]. The classes are templates for designing objects and can be re-used in developing a variety of other applications. According to [41], this presents an opportunity of assembling error-free software products from the pre-tested individual components. In addition, using these components in developing applications saves cost, time and enhances

productivity, reliability and maintainability. Recent developments in technology and coding tools have increased the capability of designing applications from reusable components. The practice is further enhanced by availability of certain ready to use, off-the-shelf software elements. The goal of component-based software engineering is to provide support in assembling the parts that are meant for reusability in future and to increase maintenance and upgradability of systems by providing customization and replacement of components [30].

The model faces challenges when assembling different components written in different programming languages requiring interoperability conversion. Again, there exists no universally acceptable framework for component-based software development. System developers using this model also face a challenge in selecting the necessary components from a multitude of them. The problem arises not only due to the large size of the repository but also from unfamiliar or unexpected terminologies employed. To facilitate the search, it is desirable to organize the components in the warehouse by expressing component relationships. Such relations allow components to be classified and understood. Finally, the model requires skills to create software architecture, test, integrate, evaluate and document products emanating from off-the-shelf software elements [18].

3.2.2 Rational Unified Process

The Rational Unified Process is a process-independent life cycle model applicable to several software engineering processes [2]. The process is iterative and incremental in nature and employs the use-case tools of the unified modeling language (UML) to specify and design a system. UML is a visual modeling language that provides a method of stipulating, constructing and documenting a project [32]. The model involves refinements of a basic model through multiple cycles while accommodating new requirements and resolving risks. It emphasizes on a robust software model not prone to failure and revision. The process is object-oriented in nature and gathers information by understanding how the delivered software is to be used and can be tailored to the needs of both small and large projects. [30] described a rational unified process with four development phases grouped under two broad categories. The engineering category made up user requirements gathering or analysis and design stages; and the production category comprising of coding/testing and deployment tasks. [32] observed that the model has a limitation in that risks are greater towards completion of a software product and it may be costly to reverse mistakes of the preceeding phases.

3.2.3 Win-Win Spiral Model

The win-win spiral model uses theory W (win-win) to develop a system that is negotiated from all the stakeholders for it to be a successful [35]. The approach builds on the normal spiral development life cycle where customers and application developers enter into a process of negotiation [36]. The consumer proposes system features, performance and other characteristics against cost and delivery time. The resulting negotiated "win-win" solution addresses most of the customer's needs and software engineers win by working under realistic budget and timelines. There are three multi-stakeholders feedback and collaboration activities: identification of the systems key consumers, determination of the stakeholders' ideal system vision (win-condition) and negotiation of the consumers' perfect system view to bring together a set of win-win condition for all the concerned groups [6]. The model implements a software product in a series of iterative phase as shown in figure 1 adapted from [7].

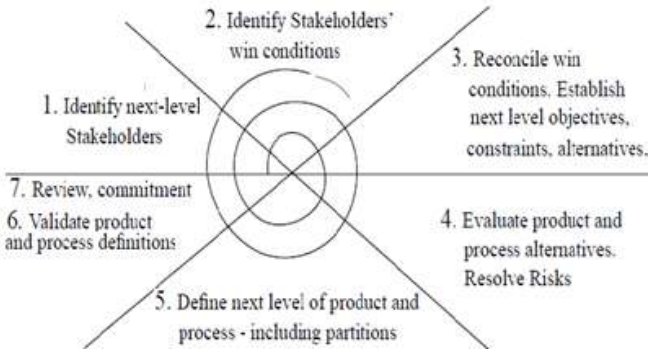

Figure 1: The Win-Win Spiral Process model

According to [35], the win-win spiral model has a higher consumer approval rate and requires less remedial work since it embraces all stakeholders' views. Additionally, the model has a higher rate of risk avoidance and greater documentation control. Lastly, the approach realizes applications early in the life cycle while giving room for inclusion of emerging functionalities later to the system. Conversely, the model can be expensive, requires expertise and its success is based, to a large extent, on the risk identification and mitigation aspects. There is also, the challenge of growing user-requirements due to stakeholders' involvement in the development process [6].

3.2.4 Rapid Application Development (RAD) Model

The model combines the benefits of iterative and incremental approaches in the process of application development. According to [29], the approach is useful in situations where user requirements are not known upfront so that a working model of the system is developed to assist the client have a physical feel of what they want. Object-oriented approach, reusable software components and modern programming tools have helped system developers to produce prototypes more speedily than using conventional models. Rapid application development describes this method of creating feasible accomplishments in a very short period of time [22]. It is characterized by user involvement in all the phases of the life cycle and close collaboration with project specialists. [30], proposed a RAD model with four phases: generation of user requirements accelerated by joint application design (JAD), user description of the application captured with the help of automated tools, construction of the prototype and implementation combined with testing and user training.

The model has several advantages such as: developing applications more quickly, considerable cost saving, encouraging stakeholders' feedback, increasing reusability of components and addressing software integration issues [34]. However, it poses challenges in its dependence on strong teams and individual performance to identify client requirements and has a narrow application to only projects that can be split into modules. It also, requires software

engineers with modeling skills and whose cost is high. Moreover, the users may continue requesting for enhancements of prototypes in search of perfect applications leading to project scope creep and delay. Finally, customers may be deceived by a prototype to imagine that the developer has completed the application leading to premature termination of services and hence, sub-standard applications [16].

3.2.5 Cleanroom Software Engineering

The model is based on perfecting the application development process to arrive at a final product that does not require reworking or costly defect removal process. The "cleanroom" environment is achieved by developing the software product in an incremental manner. The design, development and verification of each increment are undertaken in a rigorous manner based on specified, structured procedures and principles of certification and standardization [33]. The cleanroom approach anchors on five key pillars: formal specification of requirements; incremental development; structured programming; static verification of individual builds; and statistical testing of the application with the help of reliability growth models.

The model uses a method known as box structure specification where a 'box' contains aspects of a system. Information contained in each box is reliable to describe its fine-tuning, independent of other boxes implemented. According to [30], the model prescribes three boxes: black box, state box and clear box. The black box presumes the behavior of a system and it reacts to specific occurrences by applying a set of transition rules while state box encapsulates data and methods just like other objects maintained in all transitions. Finally, the clear box defines the transition function and includes the procedural design for the box. Cleanroom modeling process is advantageous in that error-free, high quality applications are developed, in shorter time and at lower costs. However, the model is deemed to be too theoretical and mathematical hence complicated and it lacks unit testing leading to low approval levels among the stakeholders [45]. Figure 2 below represents a model of the cleanroom process adapted from the description given by [44].

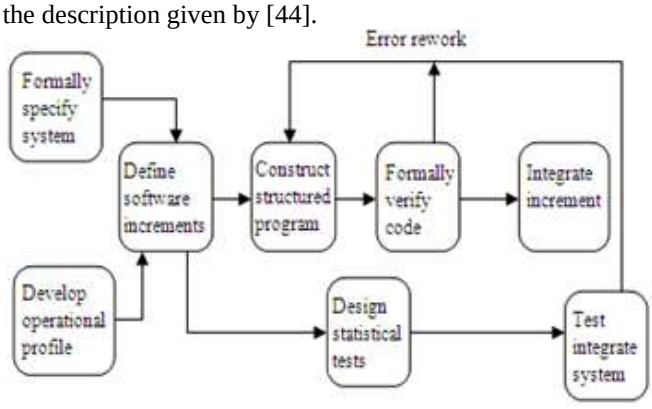

Figure 2: The cleanroom Process

3.2.6 Concurrent Development Model

'The approach, also known as concurrent engineering, is a series of software engineering tasks and their associated states [46]. In large software development projects, several engineering teams are involved and numerous activities are undertaken, simultaneously, which are at different completion stages. In addition, various activities can be in one of the several states namely: not started, commenced, being reviewed or completed [30]. This makes it difficult to keep track of the status of each activity compounded by the fact that events emanating within an activity or elsewhere can cause its transition from one state to another.

According to [46], the model can be applied to all types of application development projects since it is easy to understand and gives instant feedback after testing while providing an instant view of the project status. Conversely, there is the challenge of proper communication among the team members and requires constant recollection of the status of different activities. Figure 3 below illustrates a graphic representation of one application development motion within the modeling activity for the concurrent process model adapted from [20].

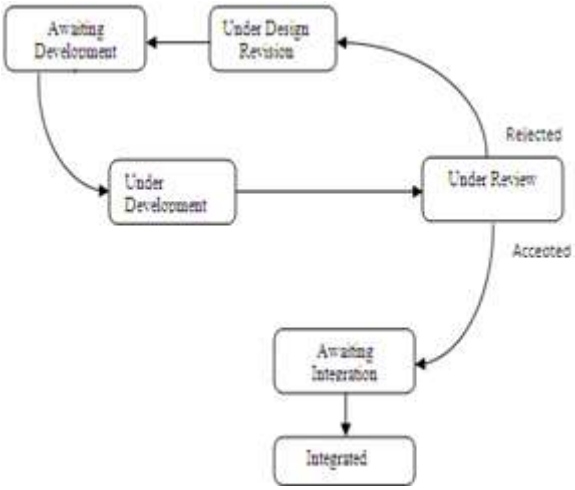

Figure 3: Activity Transition Diagram of the Concurrent Engineering Model

3.2.7 Agile Development Process

Agile development model commences by a brief recording of an application requirements in clients own words, known as user stories. Test cases are generated from the resulting software project specifications from which programmers design user interfaces. The design may then, be reworked to match the tests and user interfaces [30]. Extreme programming (XP) and scrum are the two forms of the agile development process [14]. [19], describes extreme programming as an application development process based on simplicity, communication, feedback and courage. The development team is brought together in an environment of simple practices and prenty of user feedback necessary for identifying the present status and hence allowing developers to focus their effort based on the prevailing unique status. On the other hand, scrum is a software development process that supports collaboration and responsibility in iterative advancement manner, towards a precise goal. It is grounded on starting with what is known and then, following the progress while eliminating the unnecessary elements based on three key pilllars: transparency, inspection and adaptation [31]. Agile development process is useful in situations where user requirements keep on mutating. Figure 4 represents an agile development model tailored from the descriptions given by Shelly and Rosenblatt (2009).

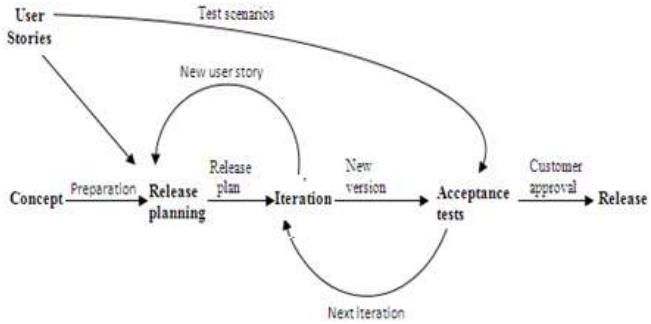

Figure 4: Agile Development Process

[15], asserts that agile development process is fast, flexible and efficient especially when implementing projects characterized by fluctuating user requirements. In addition, it fosters community values, allows constant validation of the project thereby reducing project risks and modifications. However, [17] points out several shortcomings of the model such as: lack of technical and interpersonal skills in the group, lack of laid down structures coupled with missing documentation - that can introduce an element of risk and project scope creep as user requirements continue to grow during the project. Again, there are chances that developers may fail to grasp the full picture of the system emanating from lack of quality communication with the clients. Finally, reworking the prototype is discouraging to developers and may lead to negligence of quality.

4. Emerging e-Business Application Development Models

Research has shown a growing trend of enhanced or integrated software development models with an aim of dealing with or minimizing shortcomings of the parent models or to cope with shifting user requirements and emerging trends in technology. [21], found that hybrid approaches were being used widely in software projects where the traditional models formed the backbone on which refinements hinged and then integrated to other models to generating newer software development approaches. However, their study lacked adequate population size and relied only on participants working in a company where one hybrid model was in use. Again, they employed convenience sampling technique for the study which is prone to bias leading to unreliable study findings.

[49] proposed a hybrid agile softwre development model christened Water-Srum-Fall. This was an acronym where water - represented upfront tasks of requirement generation, timelines and budget; scrum – stood for provision of simple set of values, working practices, and duties for teams to execute; and fall – stood for establishing bylaws to limit software release rates. However, software developers rarely follow the process depicted by the model. In addition, the approach poses a challenge of taking a lot of time upfront, dealing with interpersonal dynamics in agile teams and loss of clientele due infrequent releases [37].

Open source agile software development approach was coined by [28] by combining the aspects of open source software and the agile model. However, agile model has problems origianating from group dynamics, lack of documentation and standardization. Therefore, when integrated with an open source software that tend to evolve in line with developer's wishes, complicates the process further. Additionaly, open source software is not user friendly and lacks adquate testing and vendor support making the model incapable of handling emerging clients needs.

An enhanced system developmnt life cycle model was hypothesised by [27], who argued that computers could be used as persuasive tools capable of changing peoples' behavior voluntarily. In addition, the authors emphasized that inclusion of persuasion and sustainable development aspects in the conventional system development life cycle approaches would improve the models and make them more responsive. However, the study did not address the short-comings of the models like rigidity, duration of delivery and tedious documentation processes that make them unpopular among software engineers.

Many present-day software development methodologies are being used for large and complex projects. They may be used either, singly or as hybrid formats - where they are combined with other models. The models integration is aimed at addressing complex design issues encountered in a software development environment. However, disruptive technological advancements, like web 2.0 and web 3.0 platforms that are participative, collaborative and integrative in nature require more than just a hybrid of the existing models. A more focused software development method should be put in place that addresses both persistent and emerging design issues.

5. e-Business Application Development Direction

Business applications are being used by consumers with diverse expertise and abilities. Part of this development can be traced to software engineers desire to assist users in their routine activities that impact on businesses. With the advent of web 2.0 social platforms, consumers are becoming more involved in business processes such as product or service design, improvement and even distribution [23]. In addition, businesses have to cope with an equally different work force whose life is inter-twined by social media platforms. For this category of employees, there is little distinction between work-life and social-life. To enhance the productivity of these workers, firms have moved to incorporate their way of life into business operations [9].

Software engineers are faced with challenge of designing applications that cover almost all business needs while maintaining usability and understandability. They have to get input from business owners, workers and other stakeholders as well. A lot of these inputs are in non-technical forms and require extensive transformation into user requirements that can be implemented into an e-business application with fewer errors.

Interaction between e-business application and multiplicity of users, comprising of employees, producers, customers, financial institutions and others interested stakeholders, is an on-going process throughout the life of the business [9].

Such applications require a more adaptive system design methodology like the waterfall approach, modified to suit emerging technological trends. . Having the user actively involved in the process of e-business application development is a path which can improve their quality of life and give responsive side of the designer. In the end, consumers will be more contented with the resulting application with enhanced utility, rational navigation procedures and easy to use.

6. Interactive Waterfall Model

Research has demonstrated the application of the classical waterfall model in developing large and complex applications for both small and large firms. In a survey of two hundred and two specialists in Turkey, [11] found that requirements elicitation stage is the most challenging phase of the system development life cycle. [47], concurs that although agile methodologies seemed more common in the last ten years, traditional approaches, including the waterfall model, are still popular. The present study closes this gap by adopting inclusion of improved requirement elicitation task throughout the system development life cycle as proposed by [26]. In addition, this study incorporates testing in every phase of the system development life cycle as a tool for validating and verifying users input made in all the phases of the model from initiation to post-deployment stages. The study adopts waterfall model described by [48], and improves it to accomodate the web 2.0 community in software development life cycle as shown in figure 5 next.

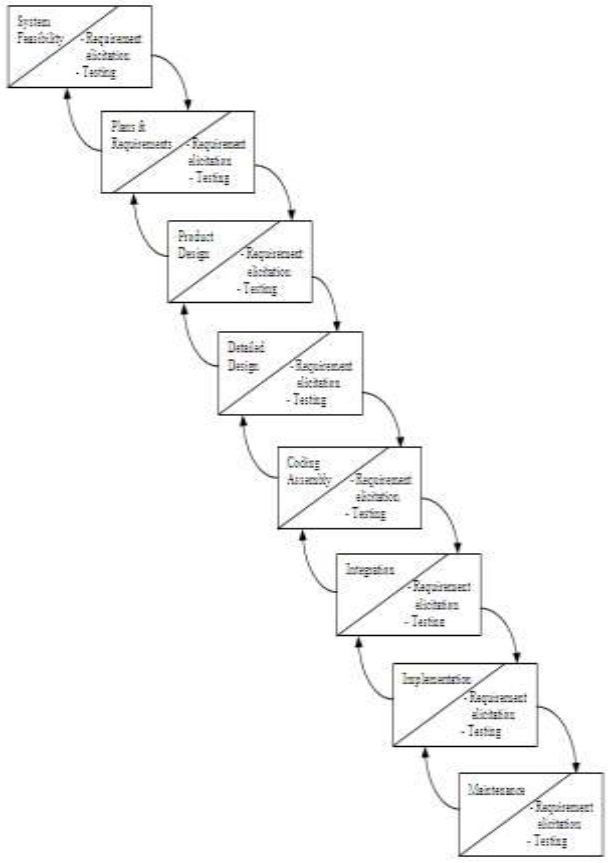

Figure 5: Interactive Waterfall Model

The model is beneficial to the e-business owners because they will be more assured that the applications developed will work as expected since user requirements have been incorporated in every phase. In addition, users will get an application with a high probability of matching their diverse work/leisure environments. Furthermore, application developers will spend little time in each phase on standards, rules and agreements already agreed upon by stakeholders making delivery of the system much faster and less costly. Finally, testing user responses at every phase for validation and verification purposes will reduce system rework and protect the business from legal liabilities that may emerge from wrong or invalid contributions. The model presents a win-win position where businesses invest in better e-business application systems and users collaborate with developers to achieve the ideal software product.

7. Conclusion

Nowadays there are numerous scientific software development models. The proposed model has been developed bearing in mind the problems faced while using the classical, contemporary and hybrid approaches in e-business application development. The proposed model is two-fold that ensures clients satisfaction by incorporating user feedback on one hand and improving the quality of the accomplished software product. Software developers employing the model will be more satisfied because reworks will be minimized and the delivery time shortened. Again, the stakeholders' acceptance of the final e-business application is higher and a motivating factor to the developers.

References

[1] Agarwal, B. B., Tayal, S. P., & Gupta, M. (2010). *Software Engineering and Testing*. Sudbury, Massachusettes: Jones and Bartlett Publishers.

[2] Akhunzada, A., Gani, A., Hussain, S., Khan, A. A., & Ashrafulla. (2015). Towards experiencing the pair programming as a practice of the Rational Unified Process (RUP). *SAI Intelligent Systems Conference (IntelliSys)* (pp. 537-542). London: IEEE.

[3] Alshamrani, A., & Bahattab, A. (2015). A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. *International Journal of Computer Science Issues, Volume 12, Issue: 1, No. 1* , 106-111.

[4] Barassi, V., & Trere, E. (2012). Does Web 3.0 come after Web 2.0? Deconstructing theoretical assumptions through practice. *New Media and Society: Vol. 18, Issue 8* , 1269-1285.

[5] Bassil, Y. (2012). A Simulation Model for the Waterfall Software Development Life Cycle. *International Journal of Engineering & Technology (iJET), ISSN: 2049-3444, Vol. 2, No. 5.*

[6] Boehm, B., Bose, P., Horowitz, E., & Lee, M. J. (1994). Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach. *Software Requirements Negotiation and Renegotiation Aids* , 1-18.

[7] Boehm, W. B. (1998). A Spiral Model of Software Development and Enhancement. *IEEE* , 61-72.

[8] Bonifati, A., Ceri, S., Fraternali, P., & Maurino, A. (2000). Conceptual Modeling for E-Business and the

Web. *Building Multi-device, Content-Centric Applications Using WebML and the W3I3 Tool Suite* (pp. ER 2000 Workshops on Conceptual Modeling Approaches for E-Business and The WorldWide Web and Conceptual Modeling). Salt Lake City, Utah, USA,: Springer.

[9] Braude, J. E., & Bernstein, E. M. (2016). *Software Engineering: Modern Approaches, 2nd Edition*. Long Crove: Waveland Press, Inc.

[10] Davis, M. A., Bersoff, H. E., & Comer, E. R. (1988). A strategy for Comparing Alternative Software Development Life Cycle Models. *IEEE Transactions on Software Engineering: Volume: 14, Issue: 10*, 1453 - 1461.

[11] Garousi, V., Coskuncay, A., Betin-Can, A., & Demirors, O. (2015). A Survey of Software Practices in Turkey. *Journal of Systems and Software, Volume 108*, 148-177.

[12] Gellersen, H. W., & Gaedke, M. (1999). Object Oriented Web Application Development. *IEEE Internet Computing, volume 3, Issue 1*, 60-68.

[13] Giardino, C., Paternoster, N., & Unterkalmsteiner, M. (2016). Software Development in Startup Companies: The Greenfield Startup Model. *IEEE Transactions on Software Engineering, Volume: 42, Issue: 6*, 585 - 604.

[14] Gregory, P., Barroca, L., Taylor, K., & Salah, D. S. (2015). Agile Processes in Software Engineering and Extreme Programming: 16th International Conference, XP 2015. *Agile Challenges in Practice: a Thematic Analysis* (pp. 64-81). Helsinki, Finland: Springer International Publishing.

[15] Heikkila, V. T., Lassenius, C., Damian, D., & Paasivaara, M. (2015). A Mapping Study on Requirements Engineering in Agile Software Development. 2015 41st *Euromicro Conference on Software Engineering and Advanced Applications* (pp. 199-207). Funchal: IEEE.

[16] Hentzen, W. (2002). *The Software Developer's Guide*. Whitefish Bay : Hentzenwerke Publishing.

[17] Käpyaho, M., & Kauppinen, M. (2015). Agile Requirements Engineering with Prototyping: A Case Study. 2015 *IEEE 23rd International Requirements Engineering Conference (RE)* (pp. 334-343). Ottawa: IEEE.

[18] Khan, S., Arif, F., Babar, M., Khan, F., & Tahir, M. (2016). Framework for Better Reusability in Component Based Software Engineering. *Journal of Applied Environmental and Biological Sciences, Volume 6*, 77-81.

[19] Kniberg, H. (2015). *Srum and XP from the Trenches: How We Do Scrum.* New York: C4Media.

[20] Kossiako, A., Sweet, N. W., Seymour, J. S., & Biemer, M. S. (2011). *Systems Engineering Principles and Practice.* New Jersey: John Wiley & Sons.

[21] Kuhrmann, M., Diebold, P., & Münch, J. (2017). Hybrid So ware and System Development in Practice:Waterfall, Scrum, and Beyond. *ICSSP 17* , 1-10.

[22] Laudon, C. K., & Laudon, P. J. (2017). *Essentials of Management Information Systems*. Mexico: Pearson.

[23] Lee, I. (2016). *Encyclopedia of E-Commerce Development, Implementation and Management*. Hershey PA: Business Science Reference.

[24] Li, W.-S., Candan, K. S., & Huang, W.-K. (2004). Acceleration and Monitoring of Data Center-hosted Distributed Database-driven Web Applications. *Concurrent Engeneering: Research and Applications Volume 12 205Number 3* , 205-219.

[25] Lu, M.-t., & Yeung, W.-l. (1998). A framework for Effective Commercial Web Application Development. *Internet Research: Electronic Networking Applications and Policy. Volume 8 • Number 2* , 166–173.

[26] Marea, D., Bulander, R., Kruslin, C., Shishkov, B., & Sinderen, V. M. (2012). e-Business Challenges and Directions: Important Themes from the First ICE-B Workshop. *ICETE* (pp. 3-35). Berlin: Springer.

[27] Mastaquim, M. M., & Nystrom, T. (2015). A System Development Life Cycle for *Persuasive Design for Sustainability. Persuasive Technology: 10th Internationl Conference, PERSUASIVE 2015 (pp. 217-230).* Chicago: Springer International Publishing.

[28] Misra, C. S., & Singh, V. (2015). Conceptualizing Open Agile Software Development Life Cycle (OASDLC) Model. *International Journal of Quality and Reliability Management, Vol 32*: Issue 3 , 214-235.

[29] Misra, S., Omorodion, M., Mishra, A., & Fernandez, L. (2017). A Proposed Pragmatic Software Development Process Model. In I. Global, *Intelligent Systems: Concepts, Methodologies, Tools and Applications* (pp. 448-462). Hershey: IGI Global.

[30] Mohapatra, K. J. (2010). *Software Engineering (A Lifecycle Approach).* New Delhi: New Age Internationa (P) Ltd., Publishers.

[31] Morris, D. (2017). *Scrum in Easy Steps: An Ideal Framework for Agile Projects*. WarwickShire: In Easy Steps Ltd.

[32] O'Regan, G. (2017). *Concise Guide to Formal Methods: Theory, Fundamentals and Industry Applications.* Gewerbestrasse: Springer International Publisher.

[33] O'Regan, G. (2017). *Concise Guide to Software Engineering: From Fundamentals to Application Methods.* Gewerbestrasse: Springer International Publishing .

[34] Patel, M. (2017). *Professional Knowledge for IBPS/SBI Specialist IT officer Exam*. New Delhi: Disha publishers.

[35] Rajiv, C. (2016). *Web Engineering*. New Delhi: Asoke K. Ghosh, PHI Private Learning Ltd.

[36] Rajput, S., & Sharma, R. (2015). *Software Engineering*. Gwalior: Horizon Books.

[37] Rastogi, V. (2015). Software Development Life Cycle Models - Comparison, Consequences. *International Journal of Computer Science and Information Technologies, Vol. 6, Issue 1* , 168-172.

[38] Rudman, J. R. (2010). Incremental risks in Web 2.0 applications. *The Electronic Library, Vol. 28 Issue: 2* , 210-230.

[39] Rudman, R. J. (2009). Incremental risks in Web 2.0 applications. *Journal of Knowledge Management Vol. 13 Issue 1* , 120-134.

[40] Rudman, R., & Steenkamp, L. (2009). Potential influence of Web 2.0 usage and security practices of online users on information management. *South Africa Journal of Information Management, Volume 11 Issue 2*, 1-13.

[41] Sarcar, V. (2016). *Interactive Object Oriented Programming in Java: Learn and Test Your Skills*. Bangalore, Kanataka: Apress.
[42] Schwinger, W., Retschitzegger, W., & Schauerhuber, A. (2008). A survey on web modeling approaches for ubiquitous web applications. *International Journal of Web Information Systems, Vol. 4 Issue : 3* , 234-305.
[43] Shang, S. S., Li, E. Y., Wu, Y.-L., & Hou, O. C. (2011). Understanding Web 2.0 service models: A knowledge-creating perspective. *Information & Management, volume 48* , 178-184.
[44] Sommerville, I. (2015). *Software Engineering 8th Edition*. Edinburg Gate: Addison Wesley Longman Limited.
[45] Stephens, R. (2015). *Beginning Software Engineering*. Indianapolis: John Wiley & Sons, Inc.
[46] Stjepandić, J., Verhagen, J. C., & Wognum, N. (2015). *Concurrent Engineering in the 21st Century: Foundations, Developments and Challenges*. New York: Springer International Publishing.
[47] Vijayasarathy, R. L., & Butler, W. C. (2016). Choice of Software Development Methodologies: Do Organizational, Project and Team Characteristics Matter? *IEEE Software, Volume 33* , 86-94.
[48] Wasson, S. C. (2015). *System Engineering: Analysis, Design and Development Principles*. New Jersey: John Wiley and Sons, Inc.
[49] West, D. (2011). *Water-Scrum-Fall Is The Reality Of Agile For Most Organizations* Today. Cambridge: Forrester Research, Inc.