A SOFTWARE SYSTEM DEVELOPMENT LIFE CYCLE MODEL FOR IMPROVED STUDENTS' COMMUNICATION AND COLLABORATION

Abdullahi Yusuf Egwoh[1],Ogwueleka Francisca Nonyelum[2]

[1,2]Computer science Department, Nigerian Defence Academy, Kaduna.

ABSTRACT

Software engineering provides methodologies, concepts and practices, which are used for analyzing, designing, building and maintaining the information in a software industry. Software Development Life Cycle (SDLC) model is an approach used in the software industry for the development of various size projects: small scale projects, medium scale projects and large scale projects. A software project of any size is developed with the co-ordination of development team. It is therefore important to assign resources intelligently to the different phases of the software project by the project manager. This study proposes a model for the spiral development process with the use of a simulator (Simphony.NET), which helps the project manager in determining how to increase the productivity of a software firm with the use of minimum resources (expert team members). This model increase the utilization of different development processes by keeping all development team members busy all the time, which helps in decreasing idle and waste time.

KEYWORDS:

Software System Development Life Cycle, Students, communication, collaboration, Software Engineering.

1. INTRODUCTION

In a software industry, software projects of various sizes are developed by using many development principles and approaches. Similarly, SDLC models are such type of methodologies which are used for the development of software project with their different development phases like analysis phase, designing phase, programming phase, testing phase and maintenance phase. In effect, many SDLC models are deeply investigated and studied by many practitioners in the world. The waterfall model, spiral model, incremental model, rapid prototyping model and agile model are few most successful SDLC models. Many software development industries adopted Spiral model as their prime development approach for the maintaining, designing, planning and programming of the software projects (Munassar, 2010). Each Spiral model's phase is handled by the team of expert employees, for example, business analysis department; software coding and programming department and software maintenance department. However, assigning the appropriate and exact number of expert team members (resources) for each phase of the spiral model is a very confusing work for the project manager of a software firm. In order to increase or maximize the productivity, it is very important to find the optimal number of resources that should be assigned to each phase of the spiral model and completes a particular phase or task. This is the reason why simulation for the SDLC model is required in order to determine the appropriate number of expert team members which are necessary to fulfil a certain project of a certain scale. Relatedly, a simulation of a system is the operation of a model of the system (Osman et al, 1990). The model can be reconfigured, studied, experimented and analyzed properly. Simulation is used before an existing system is altered or a new system built to reduce

DOI:10.5121/ijcses.2017.8401

the chances of failure, to eliminate unforeseen bottlenecks, to prevent over-utilization of resources, and to optimize the performance.

Software development life cycle (SDLC) is important for the software project success, the good software engineer should have experience and knowledge to prefer one model than another based on the project context. In order to choose the right SDLC model according to the specific concerns and requirements of the project.
In this paper, we will explore the different types of models, merits and demerits and when to use them.

2. TYPES OF SOFTWARE DEVELOPING LIFE CYCLES (SDLC)

2.1 WATERFALL MODEL

The waterfall model is a linear sequential flow in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of software implementation. This means that any phase in the development process begins only if the previous phase is complete. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement. The waterfall approach is the earliest approach and most widely known that was used for software development. Projects which not focus on changing the needed requirements. For example, projects initiated from request for proposals (RFPs), the customer has a very clear documented requirements (Melsatar, 2012).

2.2 V-SHAPED MODE

It is an extension of the waterfall model. Instead of moving down in a linear way, the process steps are bent upwards after the implementation and coding phase, to form the typical V shape. The major difference between v-shaped model and waterfall model is the early test planning in the v-shaped model. Software requirements clearly defined and known, Software development technologies and tools is well-known (Melsatar, 2012).

2.3 PROTOTYPING MODEL

It refers to the activity of creating prototypes of software applications, for example, incomplete versions of the software program being developed. It is an activity that can occur in software development. It is used to visualize some component of the software to limit the gap of misunderstanding the customer requirements by the development team. This also will reduce the iterations which may occur in waterfall approach and hard to be implemented due to the inflexibility of the waterfall approach. So, when the final prototype is developed, the requirement is considered to be frozen. It has some types, such as throwaway prototyping.

Prototypes that are eventually discarded rather than becoming a part of the finally delivered software

- Evolutionary prototyping: Prototypes that evolve into the final system through an iterative incorporation of user feedback.
- Incremental prototyping: The final product is built as separate prototypes. At the end, the separate prototypes are merged in an overall design.

- Extreme prototyping: Mainly used at web applications. Basically, it breaks down web development into three phases, each one based on the preceding one. The first phase is a static prototype that consists mainly of HTML pages. In the second phase, the screens are programmed and fully functional using a simulated services layer. In the third phase, the services are implemented. This process can be used with any software developing life cycle model. This shall be chosen when you are developing a system of user interactions. So, if the system does not have user interactions, such a system does some calculations which have no prototypes (Melsatar, 2012).

2.4 SPIRAL MODEL (SDM)

It is combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. This model of development combines the features of the prototyping model and the waterfall model. The spiral model is favoured for large, expensive and complicated projects. This model uses many of the same phases as the waterfall model, in essentially the same order, separated by planning, risk assessment, and the building of prototypes and simulations. It is used in the large applications and systems which built-in small phases or segments (Melsatar, 2012).

2.5 ITERATIVE AND INCREMENTAL MODEL

It is developed to overcome the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during the development of earlier parts or versions of the system. It consists of mini waterfalls. It is used in shrink-wrap application and large system which built-in small phases or segments. Also, can be used in a system has separated components. We can start with the budget module as a first iteration and then with inventory module and so forth (Melsatar, 2012).

2.6 EXTREME PROGRAMMING (AGILE MODEL)

It is based on iterative and incremental development, where requirements and solutions evolve through collaboration between cross-functional teams. It can be used with any type of the project, but it needs more engagement from the customer and to be interactive. Also, it can be used when the customer needs to have some functional requirement ready in less than three weeks and the requirements are not clear enough (Melsatar, 2012).

3. REVIEW OF RELATED LITERATURE

Cohen et al (2010) proposed and evaluated a SDLC model which by the use of communication and collaboration among stakeholders, improved system implementation was successfully achieved. In Goparaju et al (2012), the author focused on the productivity measurements of software development team and provides techniques and models to measure the productivity. Balci (1990) provided guidelines for conducting a successful simulation study. For the complete life cycle composed of ten (10) phases, thirteen (13) credibility assessment stages and ten (10) processes. Chi (1997) introduced the Software Engineering Process Simulation (SEPS) model for the dynamic simulation of the software project development process. SEPS is a planning tool to inspect agenda, trade-offs of cost, functionality and to test the implications of different managerial strategies on a project's outcome. Youssef (2012) made a simulation model for the

Waterfall development process. This model is built using the Simphony.NET simulation tool. In this paper author's part is to help project managers in determining how to attain the maximum productivity with the minimum number of hours, expenses and workers. In Prakriti et al (2013), the authors made a comparative study between iterative waterfall and incremental software development life cycle model for improving the resources using the Simphony.NET simulation tool. All the phases consisted in the iterative waterfall and incremental model are simulated.

4. MOTIVATION AND PROBLEM DEFINITION

In 2012, an empirical study Rupa (2012) was made in an Indian software firm. According to this study, as shown in Table 1. There were many factors which play an important role in the selection of SDLC model. One of the most important factors in this study is team size. As we know a particular software industry has software development team, the team members depends on each other and works in co- ordination. The Standish group conducted a deep investigation about this problem and shows that many software projects do not deliver on budget, on time and as expected by the customer. The main reason behind this is the assignment of expert team members unintelligently to the various phases of the SDLC models. For this reason, some phases of the process model may stay idle, while other may be delayed, due to the insufficient number of resources. The proposed simulation model for Spiral SDLC helps the project manager in determining how to increase the productivity of a software firm with the use of minimum resources (Roger, 2001). This model increase the utilization of different processes by keeping all development team members busy all the time, which help in decrease idle and waste time.

Table 1: Factors affecting the software industry (Sonia et al, 2015)

S.no	Notation used	Name of Factor
1	F1	Type of project
2	F2	Size of project
3	F3	Duration of project
4	F4	Complexity of project
5	F5	Type and level of Expected risk
6	F6	Understanding level of user requirements
7	F7	Understanding level of application areas
8	F8	Involvement of customer
9	F9	Developer's experience
10	F10	Team size
11	F11	Man machine interaction
12	F12	Availability of technology and tools

4.1 STUDENTS COMMUNICATION AND COLLABORATION MODEL

Students use digital media and environments to communicate and work collaboratively, including at a distance, to support individual learning and contribute to the learning of others. Students:

a. Interact, collaborate, and publish with peers, experts, or others employing a variety of digital environments and media.

b. Communicate information and ideas effectively to multiple audiences using a variety of media and formats.

c. Develop cultural understanding and global awareness by engaging with learners of other cultures.

d. Contribute to project teams to produce original works or solve problems.

Students need to develop these skills both online and face-to-face in order to compete and lead in the 21st century. By using Edmodo to create a student personal learning network (PLN), students take tremendous strides toward meeting and exceeding this standard. Students have used Edmodo to communicate with one another, share class data and help each other on homework. The best part of our Edmodo PLN is that it often breaks the face-to-face social boundaries that sometimes hinder effective collaboration. Students communicate and collaborate with peers that they would not normally interact with or, in some cases, people they never met. It allow an increase in students giving positive feedback, encouragement, and support to one another.

One of favourite features of Edmodo is the ability to form small groups within your class. This feature have been used in many different ways to get students to engage with their peers and truly collaborate with one another on projects or ideas. Work on team building skills and the students are required to work together on inquiry based labs and projects. The feature in Edmodo has allowed students to have a space to share their ideas, lab data, and continue their discussions beyond the classroom. It is especially helpful when students are working on group projects and need a place to share documents, answer questions, or discuss a presentation.

Before using Edmodo, students had to rely on class time to communicate with one another. When students needed to get in touch with classmates, they typically used email, texting, or other social networking sites. The advantage of using Edmodo is that it keeps student communication in one place and allows the teacher to be a part of the conversation. Students also work more efficiently and they are more willing to share their documents and ideas with the whole group. The best part is that the project manager will be able to track their progress and virtually meet with them too when needed. This interaction often led to greater understanding of content, effective collaboration, and higher quality projects and ideas.

4.2 COMPUTATION SPECIFICATION

The Spiral model which moves in clock-wise direction and the final product is delivered in many incremental iterations. For the simulation of Spiral SDLC model, following specification are made: A feedback loop is used in order to include the probability of failure concept, We can use the Symphony.NET tool with the spiral model to determine the optimized expert team members, we can deliver the complete software product in five (5) increments of the spiral simulation model. And for each increment of the simulation model, necessary and optimal resources can be determined.

The software projects arrive randomly in a software firm. We divide the projects into three types: (a) Small Scale Projects (b) Medium Scale Projects and (c) Large Scale Projects. 60% projects are small scale projects, 30% projects are medium scale projects and 10% projects are large scale projects. The projects arrive randomly and the inter-arrival time of software project is fixed by

using triangular distribution function. The minimum arrival time of a software project is 20 days, maximum time is 40 days and the average time is 30 days.
Mathematically, as in equation (1)

$$f(x|a,b,c) = \begin{cases} 0 & \text{for } x < 20 \\ \frac{2(x-a)}{(b-a)(c-a)} & \text{for } 20 \leq x < 30 \\ \frac{2}{b-a} & \text{for } x = 30 \\ \frac{2(b-x)}{(b-a)(b-c)} & \text{for } 30 < x \leq 40 \\ 0 & \text{for } 40 < x \end{cases} \quad (1)$$

The Spiral model uses five SDLC phases and the number of available expert team member for each phase, at the software firm are:15 business analyst, 20 business designer, 30 business programmer, 35 business tester, 15 business maintenance men.

The following combinations of employees are present in software industry:

1. Small-scale software projects require 1 business analyst, 2 business designer, 3 business programmer, 4 business tester and 2 business maintenance.

2. Medium scale software projects require 7 business analyst, 10 business designers, 15 business programmers, 25 business testers and 10 business maintenance.

3. Large scale software projects require 15 business analysts, 20 business designer, 30 business programmers, 35 business testers and 15 business maintenance men.

The time duration of each phase of the model to be completed is defined as follows:

A. For small scale projects

1. The lower limit of 2 days, upper limits of 3 days with uniform distribution are for business analysis phase as in equation (2):

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 2 \leq x \leq 3 \\ 0 & \text{for } x < 2 \text{ or } x > 3 \end{cases} \quad (2)$$

2. The time duration for designing phase is simulated by using exponential distribution, with mean = 2.
3. The time duration for programming phase is simulated by using exponential distribution, with mean = 4.
4. The lower limit of 6 days and an upper limit of 10 days with uniform distribution are for business testing phase as in equation (3):

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 6 \leq x \leq 10 \\ 0 & \text{for } x < 6 \text{ or } x > 10 \end{cases} \quad (3)$$

5. The lower limit of 2 days and an upper limit of 3 days with uniform distribution are for business maintenance phase. The equation is same as equation (2).

B. For medium scale projects

1. The lower limit of 5 days, upper limit of 10 days with uniform distribution are for business analysis phase as in equation (4):

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 5 \leq x \leq 10 \\ 0 & \text{for } x < 5 \text{ or } x > 10 \end{cases} \quad (4)$$

2. The duration (time) for designing phase is simulated by using exponential distribution, with mean = 8.
3. The duration (time) for programming phase is simulated by using exponential distribution, with mean = 15.
4. The lower limit of 10 days and an upper limit of 20 days with uniform distribution are for business testing phase as in equation (5):

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 10 \leq x \leq 20 \\ 0 & \text{for } x < 10 \text{ or } x > 20 \end{cases} \quad (5)$$

5. The lower limit of 7 days and an upper limit of 10 days with uniform distribution are for business maintenance phase as in equation (6):

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 7 \leq x \leq 10 \\ 0 & \text{for } x < 7 \text{ or } x > 10 \end{cases} \quad (6)$$

C. For large scale projects

1. The lower limit of 10 days and an upper limit of 20 days with uniform distribution are for business analysis phase. The equation is same as equation (5).
2. The time duration for designing phase is simulated by using exponential distribution, with mean = 25.
3. The time duration for programming phase is simulated by using exponential distribution, with mean = 35.
4. The lower limit of 30 days and an upper limit of 35 days with uniform distribution are for business testing phase as in equation (7):

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 30 \leq x \leq 35 \\ 0 & \text{for } x < 30 \text{ or } x > 35 \end{cases} \quad (7)$$

5. The lower limit of 15 days and an upper limit of 20 days with uniform distribution are for business maintenance phase as in equation (8):

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 15 \leq x \leq 20 \\ 0 & \text{for } x < 15 \text{ or } x > 20 \end{cases} \quad (8)$$

Each phase of the model, for different size projects upon completion show the following probability of having an error: for small scale project = 5%, for medium scale project = 10% and for large scale project = 35%.

5. THE SIMULATION MODEL FOR SPIRAL SDLC

The proposed model consists of many modeling elements like capture, release, task, file, counter and resource. For this model resources are the expert team members assigned on different phases of the spiral model. The simulation model for different phases of Spiral model is shown in Figure.1

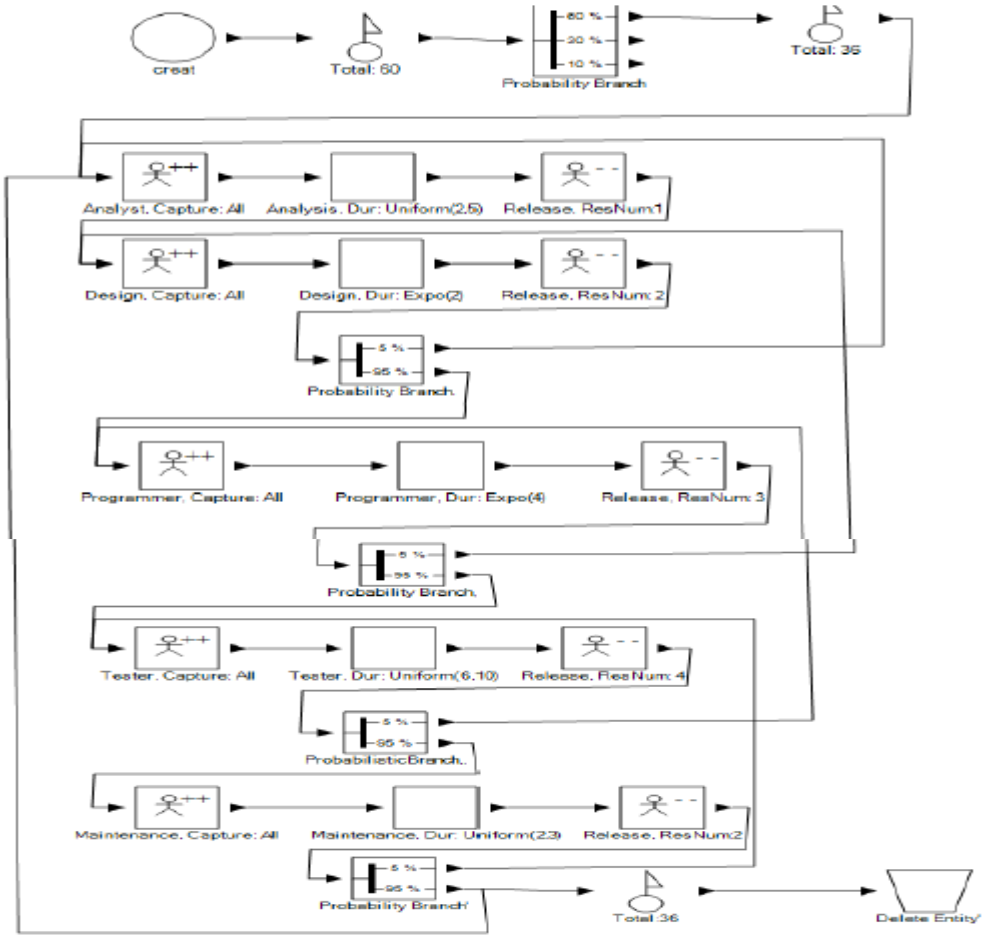

Figure 1 Simulation model for small scale projects (Sonia et al, 2015)

At very beginning of the model, a counter element counts the total number of projects received. At the end of the simulation model another counter counts the number of delivered projects. There are total of 60 projects arrive randomly in a software firm. A probabilistic branch element exists at beginning of the model with 3 core branches: branch 1 shows that 60% projects are of small scale, branch 2 shows that 30% projects are of medium scale and branch 3 shows that 10% projects are of large scale. Each phase of the model is modelled with a capture element and a release element. The capture element links a specific resource to a specific task and the release modelling element releases the resources (expert team members) from the task element when it is completed.

5.1 SIMULATION RESULTS & ANALYSIS

A. Statistics obtained for simulating the spiral model

Using the symphony.NET environment, we execute the simulation model five (5) times with 60 incoming projects. Following Table.2 and Table.3 shows the statistics obtained after running the model including average mean of the number projects delivered and the average utilization for each phase, for each increment of different scale projects.

Table.2 The Average Mean for the simulation model (Sonia et al, 2015)

Type of projects delivered	Number of projects	Average Mean
Small scale project	36	52.765
Medium scale project	18	50.765
Large scale project	6	46.765

Table.3 The Average Utilization for different scale projects (Sonia et al, 2015)

Resource name	Average utilization for:		
	Small scale projects	Medium scale projects	Large scale projects
Business Analyst	35.8%	18.4%	26.0%
Business designer	34.7%	18.1%	26.1%
Business programmer	33.6%	16.2%	26.3%
Business tester	34.7%	18.1%	25.7%
Business maintenance	34.2%	15.6%	25.6%

6. CONCLUSION

This paper proposes a model for the spiral development process with the use of a simulating tool (Simphony.NET). This model helps the software project manager in determining how to increase the productivity of a software firm with the use of minimum resources. We can assume three size projects: small scale projects, medium scale projects and large scale projects. A software project of any size is developed with the co-ordination of development team. Therefore it is important to assign team members intelligently to the different phases of the software project by the project manager. This model increase the utilization of different development processes by keeping all development team members busy all the time, which helps in decrease idle and waste time.

REFERENCES

[1] Balci, O (1990) "Guidelines for successful simulation studies,"Proceedings of the Winter Simulation Conference, Vol 1,No 1 pp. 25-32.

[2] Chi Y. Lin, Tarek Abdel-Hamid, Joseph S. Sherif (1997) "Software-Engineering Process Simulation Model (SEPS)," Journal of Systems and Software,Vol.38, No. 3, pp.263-277.

[3] Cohen. S, D.Dori and De Haan .U(2010) "A Software System Development Life Cycle Model for Improved Stakeholders Communication and Collaboration," International Journal of Computers, Communications & Control, Vol 5, No1 pp.20-41.

[4] Goparaju Purna Sudhakara, Ayesha Farooq and Sanghamitra Patnaik (2012) "Measuring Productivity of Software Development teams,"Serbian Journal of Management, Vol 7 No 1 pp. 65 – 75.

[5] https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/

[6] https://blog.edmodo.com/2011/06/02/a-model-for-student-communication-and-collaboration/

[7] Munassar. N and Govardhan. A (2010) "A Comparison between five Models of Software Engineering," IJCSI International Journal of Computer Science Issues, Vol. 7, No. 5, pp.1694-

[8] Osman Balci, Richard E. Nance, E. Joseph Demck, Emest H. Page and John L. Bishop, "Model Generation Issues in a Simulation Support Environment," in Proceedings of the I990 Winter Simulation Conference. Vol. 8, No.1, pp. 105-127.

[9] Prakriti Trivedi and Ashwani Sharma (2013) "A Comparative Study between Iterative Waterfall and Incremental Software Development Life Cycle Model for Optimizing the Resources using Computer Simulation,"Information Management in the Knowledge Economy (IMKE)Vol. 7,No.5, pp. 188-194.

[10] Roger S.Pressman,(2001) "Software engineering: A practitioner approach" published march 12, 1982, 5th ed,, macGraw-Hill, 2001.

[11] Rupa Mahanti, M.S. Neogi and Vandana Bhattacherjee(2012) "Factors Affecting the Choice of Software Life Cycle Models in the Software Industry- An Empirical Study,"Journal of Computer Science (Science Publications),Vol. 8, No. 8, pp.1253-1262.

[12] Sonia Thind, Karambir (2015)"A simulation model for spiral software development life cycle"International journal of innovative Research in computer and communication engineering Vol. 3, No. pp. 2320-2330.

[13] Youssef Bassil(2012) "A Simulation Model for the Waterfall Software Development Life cycle", International Journal of Engineering&Technology. Vol.2, no.5, pp. 2049-3444