Acknowledgement

The internship opportunity I had with BCAS campus was a great chance for learning and professional development. Therefore I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having chance to meet so many wonderful people and professionals who led me through this internship period.

I express my deepest thanks to the assessor Mr. Mohamed Ishraque, who works as an IT professional and take lectures on computer programming related subjects and others supportive subjects. He helped me taking part in useful decision & giving necessary advices and guidance and arrange all facilities for this assignment. I choose this moment to acknowledge his contribution gratefully. I acknowledge that this assignment was done with help of internet resources.

Sincerely,

S.SHALOMSHAN

Date:15.09.2022

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 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.

Contents

PART: - 1

Requirement Gathering Techniques

Brainstorming

Brainstorming is used in requirement gathering to get as many ideas as possible from group of people. Generally used to identify possible solutions to problems, and clarify details of opportunities.

Document Analysis

Reviewing the documentation of an existing system can help when creating AS–IS process document, as well as driving gap analysis for scoping of migration projects. In an ideal world, we would even be reviewing the requirements that drove creation of the existing system – a starting point for documenting current requirements. Nuggets of information are often buried in existing documents that help us ask questions as part of validating requirement completeness.

Focus Group

A focus group is a gathering of people who are representative of the users or customers of a product to get feedback. The feedback can be gathered about needs/opportunities/ problems to identify requirements, or can be gathered to validate and refine already elicited requirements. This form of market research is distinct from brainstorming in that it is a managed process with specific participants.

Interface analysis

Interfaces for a software product can be human or machine. Integration with external systems and devices is just another interface. User centric design approaches are very effective at making sure that we create usable software. Interface analysis – reviewing the touch points with other external systems is important to make sure we don't overlook requirements that aren't immediately visible to users.

Interview

Interviews of stakeholders and users are critical to creating the great software. Without understanding the goals and expectations of the users and stakeholders, we are very unlikely to satisfy them. We also have to recognize the perspective of each interviewee, so that, we can properly weigh and address their inputs. Listening is the skill that helps a great analyst to get more value from an interview than an average analyst.

Observation

By observing users, an analyst can identify a process flow, steps, pain points and opportunities for improvement. Observations can be passive or active (asking questions while observing). Passive observation is better for getting feedback on a prototype (to refine requirements), where active observation is more effective at getting an understanding of an existing business process. Either approach can be used.

Prototyping

Prototyping is a relatively modern technique for gathering requirements. In this approach, you gather preliminary requirements that you use to build an initial version of the solution - a prototype. You show this to the client, who then gives you additional requirements. You change the application and cycle around with the client again. This repetitive process continues until the product meets the critical mass of business needs or for an agreed number of iterations.

Requirement Workshops

Workshops can be very effective for gathering requirements. More structured than a brainstorming session, involved parties collaborate to document requirements. One way to capture the collaboration is with creation of domain-model artifacts (like static diagrams, activity diagrams). A workshop will be more effective with two analysts than with one.

Reverse Engineering

When a migration project does not have access to sufficient documentation of the existing system, reverse engineering will identify what the system does. It will not identify what the system should do, and will not identify when the system does the wrong thing.

Survey/Questionnaire

When collecting information from many people – too many to interview with budget and time constraints – a survey or questionnaire can be used. The survey can force users to select from choices, rate something ("Agree Strongly, agree…"), or have open ended questions allowing free-form responses. Survey design is hard – questions can bias the respondents.

Feasibility Study & Its Importance in SDLC

What is a Feasibility Study?

As the name implies, a feasibility analysis is used to determine the viability of an idea, such as ensuring a project is legally and technically feasible as well as economically justifiable. It tells us whether a project is worth the investment—in some cases, a project may not be doable. There can be many reasons for this, including requiring too many resources, which not only prevents those resources from performing other tasks but also may cost more than an organization would earn back by taking on a project that isn't profitable.

A well-designed study should offer a historical background of the business or project, such as a description of the product or service, accounting statements, details of operations and management, marketing research and policies, financial data, legal requirements, and tax obligations. Generally, such studies precede technical development and project implementation.

Understanding A Feasibility Study

Project management is the process of planning, organizing, and managing resources to bring about the successful completion of specific project goals and objectives. A feasibility study is a preliminary exploration of a proposed project or undertaking to determine its merits and viability. A feasibility study aims to provide an independent assessment that examines all aspects of a proposed project, including technical, economic, financial, legal, and environmental considerations. This information then helps decision-makers determine whether or not to proceed with the project.

The feasibility study results can also be used to create a realistic project plan and budget. Without a feasibility study, it cannot be easy to know whether or not a proposed project is worth pursuing.

Types of Feasibility Study

A feasibility analysis evaluates the project's potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. There are five types of feasibility study—separate areas that a feasibility study examines, described below.

1. **Technical Feasibility**

   This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn't want to try to put Star Trek's transporters in their building—currently, this project is not technically feasible.

2. **Economic Feasibility**

   This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

3. **Legal Feasibility**

   This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws, data protection acts or social media laws. Let's say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization's ideal location isn't zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

4. Operational Feasibility

This assessment involves undertaking a study to analyze and determine whether—and how well—the organization's needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development.

5. Scheduling Feasibility

This assessment is the most important for project success; after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete.

When these areas have all been examined, the feasibility analysis helps identify any constraints the proposed project may face, including:

- Internal Project Constraints: Technical, Technology, Budget, Resource, etc.
- Internal Corporate Constraints: Financial, Marketing, Export, etc.
- External Constraints: Logistics, Environment, Laws, and Regulations, etc.

Importance of Feasibility Study

The importance of a feasibility study is based on organizational desire to "get it right" before committing resources, time, or budget. A feasibility study might uncover new ideas that could completely change a project's scope. It's best to make these determinations in advance, rather than to jump in and to learn that the project won't work. Conducting a feasibility study is always beneficial to the project as it gives you and other stakeholders a clear picture of the proposed project.

Below are some key benefits of conducting a feasibility study:

- Improves project teams' focus
- Identifies new opportunities
- Provides valuable information for a "go/no-go" decision
- Narrows the business alternatives
- Identifies a valid reason to undertake the project
- Enhances the success rate by evaluating multiple parameters
- Aids decision-making on the project
- Identifies reasons not to proceed

Apart from the approaches to feasibility study listed above, some projects also require other constraints to be analyzed -

- Internal Project Constraints: Technical, Technology, Budget, Resource, etc.
- Internal Corporate Constraints: Financial, Marketing, Export, etc.
- External Constraints: Logistics, Environment, Laws, and Regulations, etc.

Benefits of a Feasibility Study

Preparing a project's feasibility study is an important step that may assist project managers in making informed decisions about whether or not to spend time and money on the endeavor. Feasibility studies may also help a company's management avoid taking on a tricky business endeavor by providing them with critical information.

An additional advantage of doing a feasibility study is that it aids in the creation of new ventures by providing information on factors such as how a company will work, what difficulties it could face, who its competitors are, and how much and where it will get its funding from. These marketing methods are the goal of feasibility studies, which try to persuade financiers and banks whether putting money into a certain company venture makes sense.

What Is Included in a Feasibility Study Report?

When starting a business, one of the most important steps is to conduct a feasibility study. This study will help to determine if your business idea is viable and has the potential to be successful. Several factors need to be considered when conducting a feasibility study, including the marketability of your product or service, the competition, the financial stability of your company, and more. A feasibility study should cover the amount of technology, resources required, and ROI.

The results of your feasibility studies study are summarized in a feasibility report, which typically comprises the following sections.

- Executive summary
- Specifications of the item or service
- Considerations for the future of technology
- The marketplace for goods and services
- Approach to marketing
- Organization/staffing
- Schedule
- The financial forecasts
- Recommendations based on research

Tools for Conducting a Feasibility Study

Suggested Best Practices

While every project has its own goals and needs, the following are best practices for conducting a feasibility study.

- Do a preliminary analysis. This includes getting feedback from relevant stakeholders on the new project. Also, look for other business scenarios.
- To ensure that the data is solid, determine and ask queries about it in the initial phase.
- Take a market survey to identify market demand and opportunities for the new concept or business.
- Create an organizational, operational, or business plan. This includes identifying how much labor is required, what costs, and how long.
- Make a projected income statement that involves revenue, operating expenses, and profit.
- Create an opening day balance sheet.
- You will need to identify and address any vulnerabilities or obstacles.
- Take an initial decision to go ahead with the plan.

Suggested Components

- Here are some suggested components for conducting a feasibility study:
- Executive Summary: Write a narrative describing the project, product, or service.
- Technological considerations: Ask yourself what it will take. Are you able to afford it? How much will it cost?
- Current marketplace: Find out the market for your product, service, or plan in the local and global markets.
- Marketing strategy: Define in the detailed description.
- Required staff: What human resources are needed for this project?
- Timeline and schedule: Use important interim markers to indicate when the project will be completed.
- Project financials. Project financials are the different ways managers can account for money spent and earned on projects. One of the most important aspects of financial management is creating and tracking accurate project financials.

Examples of a Feasibility Study

A local university was concerned about the state of the science building, which was built in the 1970s. School officials sought to determine the costs and benefits of expanding and upgrading the building, given the scientific and technological advances over the past 20 years. A feasibility study was therefore conducted.

School officials looked at several options and weighed the costs and benefits of updating and expanding the science building. There were concerns expressed by school officials about the project's cost and public reaction. The proposed new science building will be larger than the current one. The community board rejected similar proposals in the past. The feasibility study will address these concerns and any possible legal or zoning issues.

The feasibility study examined the technology requirements of the proposed concept (new science building), the potential benefits for students, and its long-term viability. Modernizing the science facility will increase the scientific research potential and ameliorate its modules. It also would allure new students.

Financial projections provided information about the scope & cost of this project and also provided information on raising funds. These covers issuing an investor's bonds and tapping into its endowment. Projections also help determine how the new science program attracts more fresh students to enroll in offered programs, increasing tuition and fees revenue.

The feasibility study proved that the proposed concept was feasible, which allowed for the expansion and modernization of the science building. The feasibility study would not have allowed school administrators to know if the expansion plans were feasible without it.

What is the Purpose of a Feasibility Study?

A feasibility study is an important first step in starting a new business. It is a detailed examination of whether or not a proposed business venture is likely to be successful. A feasibility study aims to provide information that will help business owners make informed decisions about their new venture.

The feasibility study will answer important questions about the proposed business, including:

- What is the target market for this business?
- Who are the competitors?
- What are the costs associated with starting and running this business?
- What are the potential risks and rewards associated with this venture?
- How much revenue can this business generate?
- What are the estimated profits and losses for this business?
- What is the potential for growth in this industry?

How Do You Write a Feasibility Study?

This feasibility study will outline why your business idea is worth pursuing and will also help you identify any potential risks or problems that could occur. When writing a feasibility study, there are a few key things to keep in mind:

1. Outline your target market and how you plan to reach them.
2. Discuss your product or service in detail and explain why it is unique and needed.
3. Outline your financial projections and explain how you plan to make a profit.

7 Steps to Do a Feasibility Study

1. Conduct a Preliminary Analysis

A preliminary investigation is necessary to determine whether a full feasibility study is warranted. During this stage, key information will be gathered to assess the project's potential and make a preliminary decision about its feasibility. This should include a review of relevant documents, interviews with key personnel, and surveys of potential customers or users.

2. Prepare a Projected Income Statement

To do a feasibility study, you must create a projected income statement. Your projected income statement will show how much money your business is expected to make in the coming year. It will include both your estimated revenue and your estimated expenses. This document will be essential in helping you make informed decisions about your business.

3. Conduct a Market Survey, or Perform Market Research

Conducting market research is an important step in any feasibility study. By understanding the needs and wants of your potential customers, you can determine if there is a market for your product or service. You can also get an idea of what your competition is doing and how to best position your business to meet the needs of your target market.

There are a variety of ways to conduct market research. One popular method is to conduct a survey. You can survey potential customers directly or use data from secondary sources such as surveys conducted by other organizations. You can also use focus groups or interviews to get feedback from potential customers.

Once you have gathered your data, you can use it to create a profile of your ideal customer. This will help you understand your target market and how to reach them.

4. Plan Business Organization and Operations

When starting a business, one of the first things you need is to plan your organization and operations. This involves creating a structure for your company and figuring out the logistics of how you will run it. There are many factors to consider when planning your organization and operations, such as:

- Company Structure: What type of company will you be (sole proprietorship, partnership, corporation, etc.)? What will the hierarchy look like?
- Location: Where will your business be located? Will you have a physical storefront or operate online only?
- Marketing: How will you promote your business?

5. Prepare an Opening Day Balance Sheet

The opening day balance sheet is a snapshot of the company's financial position at the beginning of the business venture. The purpose of the opening day balance sheet is to give an idea of the amount of money that the company has to work with and track its expenses and income as they occur. This information is vital to making sound business decisions. The opening day balance sheet will include the following:

- Cash on hand
- Accounts receivable
- Inventory
- Prepaid expenses
- Fixed assets
- Accounts payable
- Notes payable
- Long-term liabilities
- Share

6. Review and Analyze All Data

The feasibility study should include reviewing and analyzing all data relevant to the proposed project. The data collected should be verified against source documentation, and any discrepancies should be noted. The purpose of the feasibility study is to provide a basis for making a decision, and the data should be sufficient to support that decision.

The analysis should consider both the positive and negative aspects of the proposed project. The financial analysis should be thorough, and all assumptions should be documented. The risk assessment should identify any potential risks and mitigation strategies. The team assigned to the project should review the feasibility study and recommend the organization's leadership.

Organizational leadership should decide whether to proceed with the project based on the feasibility study's findings. If the project is approved, the organization should develop a project plan that includes a detailed budget and timeline

7. Make a Go/No-Go Decision

It is important to know when to cut your losses when starting a business. The go/no-go decision in a feasibility study comes in. The go/no-go decision is a key part of a feasibility study, and it can help you determine whether or not your business idea is worth pursuing.

Making the go/no-go decision is all about risk assessment. You need to weigh the risks and rewards of starting your business and decide whether the potential rewards are worth the risks. If the risks are too high, you may want to reconsider your business idea.

How to Conduct a Feasibility Study

Now, let's discuss a few of the steps we take in order to do the feasibility study.

- To begin, we do a preliminary study of the business case to define what is included and what we are examining and attempting to find is realistic.
- Following that, we generate a forecasted income statement. We need to understand the revenue sources; how are we going to profit from this? Where does the income originate? Additionally, we must do a market study.
- We need to find out whether this is a demand for our product. How much demand does this have? Is there a market for this product or service?
- Plan your company's structure and operations, which is the fourth step. Specifically, what type of organization do we need, and what resources do we have? Do we have any specific personnel needs?
- We also plan to generate a balance sheet on the first day. What are the income and expenses, and how can we be confident we'll be able to decide whether we're going to make our ROI?
- As a result, we plan to go through and examine all of our data before making a final decision on whether or not to go forward. In other words, are we going to pursue this project or business opportunity?

PART: - 2

Predictive vs. Adaptive SDLC: What is the Difference?

Predictive Software Development Life Cycle: An Overview

As the name suggests, predictive SDLC assumes you can predict the complete workflow. It involves fully understanding the final product and determining the process for delivering it. In this form of project life cycle, you determine the cost, scope, and timeline in the early phases of the project.

One of the most common predictive models is the waterfall model. It assumes various phases in the SDLC that can occur sequentially, which implies that one phase leads into the next phase. In simple words, in waterfall model, all the phases take place one at a time and do not overlap one another.

While the waterfall model is quite simple and easy to use and understand, it also entails a few drawbacks that could drastically impact your project.

Since the waterfall model follows a sequential approach, once an application is in the testing phase, it becomes difficult to go back and debug it in the development stage.

.

Pros of Predictive SDLC

- It is easy to understand and follow as each phase is initiated after another phase is completed.
- The laid down instructions and concise workflow makes it easier for the developers to work within a specified budget and timeframe.
- It enables organizations to assume the expected project budget and timelines (IF all goes as planned).
- Each stage in the predictive SDLC has specific timelines and deliverables, which makes it easier for teams to operate and monitor the entire project.

Cons of Predictive SDLC

- Working software is produced at a later stage in predictive SDLC, which leads to delayed identification of bugs and vulnerabilities in the application.
- Organizations often have to bear additional costs of delayed applications if bugs are discovered in the testing phase of the project.
- It is not the ideal SDLC model for complex projects.
- Predictive SDLC is not suitable for dynamic projects that entail flexible requirements or uncertainty in the end product.

The main concern of a predictive SDLC approach is to develop and maintain the specifications of the final product. This makes it ideal for projects where all the requirements are defined and well understood with a clear vision of the final product.

In predictive SDLC, there are minimal expected changes as the work is already predictive and well-known. The team has a clear idea of exactly where the project is heading and how to follow the sequence.

On the other hand, a predictive approach can be extremely rigid, requiring developers to maintain strict and rigorous standards throughout the life cycle. Since the sequence of the work is already predetermined, any subsequent changes can be very costly and time-consuming.

Adaptive Software Development Life Cycle: An Overview

Adaptive SDLC approaches have a mix of incremental and iterative development. It involves adding features incrementally and making changes and refinements according to feedback. In other words, the work can easily adapt to the changing requirements based on new feedback received from the client.

Agile and other iterative methodologies fall under the umbrella of adaptive SDLC. A key element of adaptive SDLC methodologies is that while it defines certain milestones throughout the SDLC, it also allows flexibility to achieve them.

Adaptive SDLC, such as Agile, focuses on achieving the desired end goal by quickly adapting the dynamic business requirements. It puts more focus on the present requirement and leaves room for future scope of the project.

Pros of Adaptive SDLC

- Adaptive SDLC entails iterative, evolutionary and incremental methodologies which offer flexible guidelines and easy flow of work.
- Methodologies such as Agile are efficient in nature and enhance team collaboration.
- Short feedback loops lead to quick adaptation to changing requirements.
- Reduces potential vulnerabilities and bugs at the deployment stage as the application is frequently tested while in the development phase.
- It focuses on delivering high quality applications while maintaining technical excellence.
- Encourages different teams to work together on a project, increasing face-to-face interactions and building better work environments.

Cons of Adaptive SDLC

- It demands for extensive client/user involvement throughout the SDLC.
- Various teams have to work together continuously while working with adaptive SDLCs, and this involves numerous interactions. Continuous communication between teams can be time consuming and require more commitment.
- Since adaptive SDLC requires close collaboration between organizations and their clients, lack of commitment from either of the sides could impact software quality.
- Frequent changes are adopted just in time for development which might result in less detailed documentation.

Adaptive SDLC approaches are best for projects that have the potential for significant changes in scope or that there is uncertainty in what is desired. You may need to adapt to the changing demands of the client for these projects.

The adaptive SDLC methodology is typically faster than predictive SDLC approaches. This is primarily due to the fact that few projects are sufficiently understood to really use a predictive SDLC methodology. When requirements are not sufficiently understood, issues are identified late in the lifecycle and this leads to expensive re-work.

Software Development Life Cycle

SDLC

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software's. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

SDLC is the acronym of Software Development Life Cycle.

It is also called as Software Development Process.

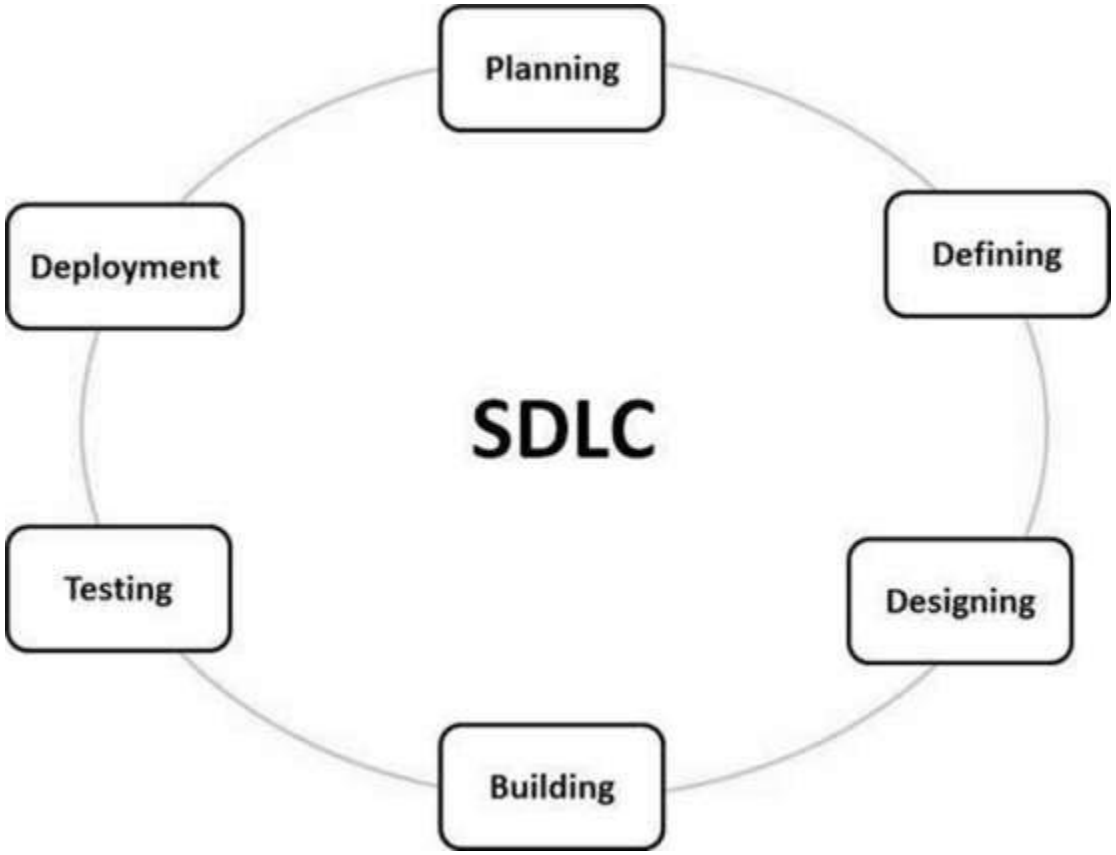
SDLC is a framework defining tasks performed at each step in the software development process.

ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

What is SDLC?

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.



Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third-party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

SDLC Models

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model

Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

SDLC - Waterfall Model

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

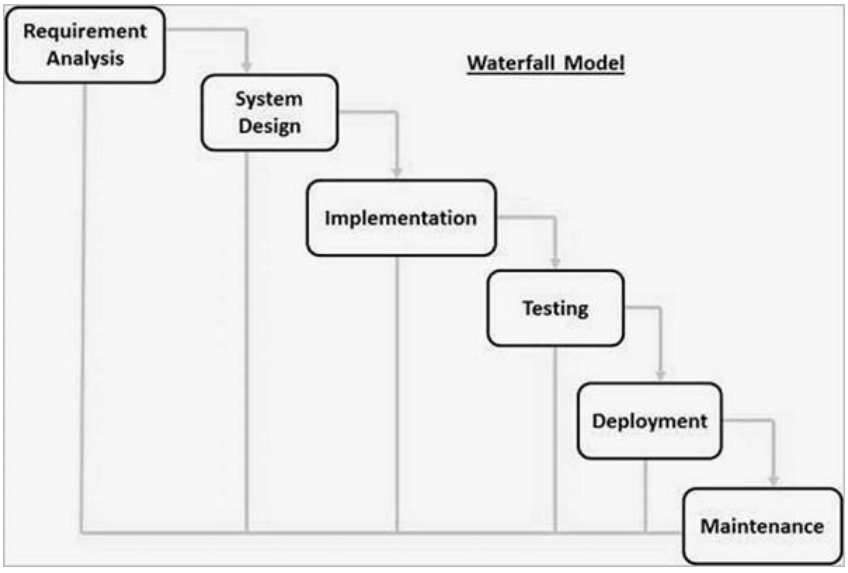
The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall Model – Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

Waterfall Model – Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model – Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.

- Easy to arrange tasks.
- Process and results are well documented.

Waterfall Model – Disadvantages

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty are high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

PART: -3

PART: -3.1

Software paradigm & Software Development Life Cycle

Software paradigm refers to method and steps, which are taken while designing the software programming paradigm is a subset of software design paradigm which is future for other a subset of software development paradigm. Software is considered to be a collection of executable programming code, associated libraries, and documentation. Software development paradigm is also known as software engineering, all the engineering concepts pertaining to developments software applied. It consists of the following parts as Requirement Gathering, Software design, Programming, etc. The software design paradigm is a part of software development. It includes design, maintenance, programming.

Software development life cycle

SDLC is the acronym for software development life cycle. It is also called the software development process. All the tasks required for developing and maintaining software. It consists of a plan describing how to develop, maintain, replace and alter the specific software. It is a process for planning, creating, testing, and information system. It is a framework of describes the activity performed at each stage of software development. It is a process used by a system analyst to develop an information system including requirements, validation, training, and ownership.

Benefits of software development life cycle

- It allowed the highest level of management control.
- Everyone understands the cost and resources required.
- To improve the application quality and monitor the application.
- It performs at every stage of the software development life cycle.

Different types of software development life cycle models

There are various software development life cycle models. These models are referred to as the software development process models. The models defined and designed which followed during the software development process.

1. **Waterfall model**

   The waterfall model is easy to understand and simple to manage. The whole process of software development is divided into various phases. The step of requirements analysis, integration, maintenance.

2. **Iterative model**

   It is repetition incarnate. In short, it is breaking down the software development of large applications into smaller pieces.

3. **Spiral model**

   It helps the group to adopt elements of one or more process models. To develop strategies that solve uncertainty and risk.

4. **V-model**

   It is known as the verification and validation model. It is characterized by a corresponding testing phase for the development stage. V model joins by coding phase.

5. **Big Bang model**

   It focuses on all types of resources in software development and coding. Small project with smaller size development team which are working together.

Stages of SDLC model

Here, we will give you a brief overview of SDLC stages as follows.

Stage-1: Requirement gathering

The feasibility report is positive towards the project and next phase start with gathering requirement from the user. Engineer communicates with the client and end-users to know their idea and which features they want to software to include.

Stage-2: Software design

It is a process to transform user requirements into a suitable form. It helps programmers in software coding. There is a need for more specific and detailed requirements in software. The output of the process can directly be used in implementation in a programming language. There are three design levels as follows.

1. **Architectural design**

   It is the highest abstract version of the system. In a software system, many components interact with each other.

2. **High-level design**

   It focuses on how the system along with all its components and its can be implemented in form of modules.

3. **Detailed design**

   It defines the logical structure of each module and its interface to communicate with each module.

Stage-3: Developing Product

In this phase of SDLC, you will see how the product will be developed. It is one of the crucial parts of SDLC, it is also called the Implementation phase.

Stage-4: Product Testing and Integration

In this phase, we will integrate the modules and will test the overall product by using different testing techniques.

Stage-5: Deployment and maintenance

In this phase, the actual deployment of the product, or you can say the final product will be deployed, and also, we will do maintenance of product for any future update and release of new features.

PART: - 4

PART: - 4.1

INFORMATION GATHERING

The first step in user requirements analysis is to gather background information about the users and stakeholders and the processes that currently take place. The following methods may be adopted: Stakeholder analysis identifies all the users and stakeholders who may influence or be impacted by the system. This helps ensure that the needs of all those involved are taken into account. If required, the system is tested by them. User groups may include end users, supervisors, installers, and maintainers. Other stakeholders include recipients of output from the system, marketing staff, purchasers and support staff (Taylor, 1990). Stakeholder analysis identifies, for each user and stakeholder group, their main roles, responsibilities and task goals in relation to the system. One of the main issues is how to trade-off the competing needs of different stakeholder groups in the new system (see 4.5 Allocation of function and user cost-benefit analysis).

Secondary market research involves researching published sources such as research reports, census data, demographic information, that throw light upon the range of possible user markets. Websites representing special groups of users such as that for the Royal National Institute for the Blind(www.mib.org.ukldigital) give information about the nature of the user population they represent (Mander & Smith, 2002).

Context of use analysis is used when a system or product is developed. The quality of a system, including usability, accessibility and social acceptability factors, depends on having a very good understanding of the context of use of the system. For example, a bank machine (ATM) will be much more usable if it is designed for use at night as well as during the day, in bright sunlight as well as normal light, and by people in wheelchairs as well as those able to stand. Similarly in an office environment, there are many characteristics that can impinge on the usability of a new software product e.g. user workload, support available, or interruptions. Capturing contextual information is therefore important in helping to specify user requirements. In order to gather contextual information, stakeholders attend a facilitated meeting, called a Context Meeting. Here

a questionnaire is completed to capture the characteristics of the users, their tasks and operating environment (see main headings in Table 1 below).

User group	Tasks	Technical environment
• System skills and experience. • Task knowledge. • Training. • Qualifications. • Language skills. • Age & gender. • Physical and cognitive capabilities. • Attitudes and motivations.	• Task list. • Goal. • Output. • Steps. • Frequency. • Importance. • Duration. • Dependencies.	• Hardware. • Software. • Network. • Reference materials. • Other equipment.
Physical environment	**Organisational environment**	
• Auditory environment. • Thermal environment. • Visual environment. • Vibration. • Space and furniture. • User posture. • Health hazards. • Protective clothing & equipment.	• Work practices. • Assistance. • Interruptions. • Management & communications structure. • Computer use policy. • Organisational aims. • Industrial relations. • Job characteristics.	

Context of use analysis was one of the outcomes of the ESPRIT HUFIT project and developed further in the ESPRIT MUSiC project (Bevan and Macleod, 1994). Context of use analysis within usability activities are also reviewed in Maguire (200Ic).

Task analysis involves the study of what a user is required to do in terms of actions and/or cognitive processes to achieve a task. A detailed task analysis can be conducted to understand the current system, the information flows within it, the problems for people, and opportunities that indicate user needs. There are many variations of task analysis and notations for recording task activities. One of the most widely used is hierarchical task analysis, where high level tasks are de-composed into more detailed components and sequences. Another method creates a flow chart showing the sequence of human activities and the associated inputs and outputs (Ericsson 2001). Kirwan & Ainsworth (1992) provide a guide to the different task analysis methods, while Hackos & Redish (1998) explain some of the simpler methods for user interface design.

Rich pictures can help stakeholders map, explore and understand a complex problem space and thereby help to identify hidden requirements (Checkland, 1981). The technique involves creating a series of sketches to show how people and systems relate to each other in an organization. They may show peoples' roles, power structures, communications and reporting mechanisms. Drawing simple figures of people with thought and speech bubbles linked to them can show particular problem areas in the current environment that may lead to new user requirements.

Field study and observational methods involve an investigator viewing users as they work and taking notes of the activity that takes place. Observation may be either direct, where the investigator is actually present during the task, or indirect, where the task is recorded on videotape by the analysis team and viewed at a later time. The observer tries to be unobtrusive during the session and only poses questions if clarification is needed. Obtaining the co-operation of users is vital so the interpersonal skills of the observer are important. For further information see Preece et al. (1994).

Diary keeping provides a record of user behavior over a period of time. They require the participant to record activities they are engaged in throughout a normal day that may lead to the identification of user requirements for a new system or product. Diaries require careful design and prompting if they are to be employed properly be participants.

Video recording can be used to capture human processes in a stakeholder's workplace or other location. The results can then be revised for the purpose of understanding more about the work and generating relevant questions relevant to user needs. Video can also be a useful supplement to other method e.g. to demonstrate new system concepts to users during user/stakeholder discussion groups.

PART: - 4.2

What is a Finite State Machine?

Introduction

A Finite State Machine, or FSM, is a computation model that can be used to simulate sequential logic, or, in other words, to represent and control execution flow. Finite State Machines can be used to model problems in many fields, including mathematics, artificial intelligence, games or linguistics.

Definition

A Finite State Machine is a model of computation based on a hypothetical machine made of one or more states. Only one single state of this machine can be active at the same time. It means the machine has to transition from one state to another in to perform different actions.

A Finite State Machine is any device storing the state of something at a given time. The state will change based on inputs, providing the resulting output for the implemented changes.

Finite State Machines come from a branch of Computer Science called “automata theory”. The family of data structure belonging to this domain also includes the Turing Machine.

The important points here are the following:

- We have a fixed set of states that the machine can be in
- The machine can only be in one state at a time
- A sequence of inputs is sent to the machine
- Every state has a set of transitions and every transition is associated with an input and pointing to a state

Real world examples

Let's see what could be a Finite State Machine in the real world:

Coin-operated turnstile

- **States:** locked, unlocked
- **Transitions:** pointing a coin in the slot will unlock the turnstile, pushing the arm of the unlocked turnstile will let the costumer pass and lock the turnstile again

Traffic Light

- **States:** Red, Yellow, Green
- **Transitions:** After a given time, Red will change to Green, Green to Yellow, and Yellow to Red

A Safe

- **States:** Multiple "locked" states, one "unlocked" state
- **Transitions:** Correct combinations move us from initial locked states to locked states closer to the unlocked state, until we finally get to the unlocked state. Incorrect combinations land us back in the initial locked state

Programming example

Knowing what we know, let's make a simple and basic programming example. Let's say we have our friend Mario in our favorite video game. So, Mario can do the following actions:

- Stand still
- Run
- Jump
- Duck

We can see that Mario can do a lot of things and all of those things should be a specific state. Before writing any single line of code, we should ask ourselves how all of our different states fit together. We need to know exactly what Mario can do and how and when he can do these actions. For example, Mario can stand still then run when we push the corresponding button, but cannot run while he is jumping. So, an input that doesn't cause a change of state is ignored.

We could start first by defining a "State Interface":

```
interfaceState
{
void                    enter(Character                    character);
State     handleInput(Character     character,     Input     input);
void                   update(Character                    character);
}
```

Then, we can a create a class for each state:

```
class RunningState : State
{
    void enter(Character character)
    {
        // Various operations like
        // setting the graphics
    }    State handleInput(Character character, Input input)
    {
        // Various operations like
        // checking the input        // Returning a state after all
the operations
```

```
        return new StandingState();
    }    void update(Character character)
    {
        // Various operations
    }
}
```

So, our character, here our dear Mario, could have the following code:

```
class Mario : Character, Player
{
    private State _state;    void handleInput(Input input)
    {
        _state.handleInput(this, input);
    }    void update()
    {
        _state.update(this);
    }
}
```

Stack-Based FSM

We could go a little further. We could use a Stack-Based FSM. With our previous solution, we have no concept of history. We know our current state, but we can't go back to the previous state.

To solve this problem, we could use a Stack, which stores elements in LIFO style (Last In, First Out), to save our different states. That means the current state is the one on the top of the Stack. Then, when we want to transition to a new state, we push that new state onto the Stack and this state becomes the current state. When we are done, we pop this state and the previous state becomes the current state. Of course, it is our responsibility to manage which state we want in the Stack and which we want to discard.

Conclusion

Through this article, we saw what a Finite State Machine is. We saw that a Finite State Machine is a model of computation based on a hypothetical machine made of one or more states and only one single state of this machine can be active at the same time.

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 increases the utilization of different development processes by keeping all development team members busy all the time, which helps in decrease idle and waste time.

Reference

[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