ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

An Innovative Research on Software Development Life Cycle Model

Sucharita Bansal[1], Ankush Goyal[2]

PG Student, Dept. of CSE, Shri Ram College of Engineering and Management, Affiliated to Maharshi Dayanand University, Palwal, India[1]

Assistant Professor, Dept. of CSE, Shri Ram College of Engineering and Management, Affiliated to Maharshi Dayanand University, Palwal, India[2]

ABSTRACT*:* The Software Life Cycle Model (SLCM) is a process of building a good software & its lifecycle stages provides Quality & Correctness of good software. All the stages of lifecycle are important in itself. One wrong step in lifecycle can create a big mistake in the development of Software.
Being an owner of Software Company we must know the development life cycle of the software. Even our buyer may also be aware of this life cycle. So, everyone wants to know that how its development begins, which are the development processes, which is the end portion of development life cycle.

KEYWORDS*:* Software, System Development Life Cycle Model (SDLC) and Project Management, Software Development Process or Activities, Stakeholders

I. INTRODUCTION

Software is basically the combination of computer programs, data structure and documentation. The Aim of the Software Engineering is to estipulate a concrete framework to build a high quality of software.

The software industry includes many different processes, for e.g., analysis, development, maintenance and publication of software. This industry includes software services, such as training, documentation and consulting [1].

- ***Programs***
  Software is a set of computer programs or instructions that provides the desired function and performance.

- ***Data Structure***
  Software is a data structure that enables the program to adequately manipulate information.

- ***Documentation***
  Documentation describes the operation and use of programs.

Hence, Software is composed of computer programs, data structure & documentation.

Software = Programs + Data Structure + Documentation

A. Difference between Software & Program?

i. Programs are developed by individual for Personal use. They are therefore, small in size & have limited functionality, but Software Products are extremely large.

ii. In case of Program, A single developer is involved and user interface may not be so important because the programmer him/herself is the sole user. But In case of Software Products, A large number of developers are involved but users who are not involved with the development are attached.

iii. A Program can be developed according to the programmer's individual style of development. But Software Products must be development using software Engineering Principles.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

II. IMPORTANCE OF SOFTWARE DEVELOPMENT PROCESS MODELS OR ACTIVITIES

A Software Life Cycle Model (SLCM) is also called a **Software Development Process Model**. System development is the process of defining, designing, testing and implementing a software application. This is a structure imposed on the development of a software product. A system development project includes all the activities from the time a potential requirement has been identified until the system has been fully implemented [2].

There are several Models for such processes, each describing approaches to a variety of tasks or activities that take place during the process. ISO/IEC 12207 is an international standard for software lifecycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

The Analysis, Designing, Implementation, Testing, Documenting, Evaluation and Maintenance are the steps in the Software life cycle development process. We have numerous types of SDLC Models like Waterfall, Agile, and Spiral etc. Several Models exist to streamline the development process [2] [3].

- **Waterfall Model**
  In this model, after each phase is completed, it proceeds to the next stage. Reviews may occur before moving to the next phase, which allow for the possibility of changes. Waterfall discourages revisiting any prior phase once it is complete. This "inflexibility" is the main limitation of this model [4].
  The waterfall model depicts a process, where developers are to adopt the following phases in order:
  1) Requirement Specification (Requirement Analysis)
  2) Software Design
  3) Integration
  4) Testing (or Validation)
  5) Deployment (or Installation)
  6) Maintenance

- **Spiral Model**
  The main characteristic of a Spiral model is risk management at regular stages in the development cycle. Barry Boehm proposed a formal software system development "spiral model", with emphasis on a key area of risk analysis that is neglected by other methodologies. The spiral model is visualized as a process passing through a number of iterations. The first stage is to formulate a plan, and then strive to find and remove all potential risks through careful analysis and, if necessary, by constructing a prototype. If some risks cannot be ruled out, the user has to decide whether to terminate the project or to ignore the risks [5].

- **Iterative and incremental development Model**
  Iterative development model prescribes the construction of initially small but ever-larger portions of a software project to help all those involved to uncover important issues early before problems or faulty assumptions can lead to disaster. Iterative processes can assist with revealing and refined definition of design goal [6].

- **Agile development Model**
  Agile software development uses iterative development as a basis but advocates a lighter and more people-centric viewpoint. Agile processes use feedback as primary control mechanism. First, one writes automated tests, to provide concrete goals for development. The next step is coding by a pair of programmers, which is complete when all the tests are successfully passed. The incomplete but functional system is demonstrated for the users. At this point, the practitioners start again on writing tests for the next most important part of the system [7].

- **Code and fix**
  "Code and fix" development is not a deliberate strategy and schedule pressure on software developers. With incomplete design, programmers begin producing code. At some point, testing begins (often late in the development cycle), to fix the bugs before shipment [8].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

III. PROPOSED WORK

A **Software Process** consisting of many different components processes. Most important being Project Management (PM) & Development Process (DP).

Project Management (PM) is a task that requires a lot of time and effort. A Software life cycle model is allowing focus on important features of the work, downplaying excessive detail. It is providing a standard work unit hierarchy for progressive decomposition of the work into manageable chunks. It also provides a framework for definition and storage of the deliverables product produced by the projects and communicates our development strategy to project stakeholders [2] [9].

All this SDLC Model must follow the steps for developing errorless software [4]. There are seven types of software development Activities that are mentioned as below:-

(i) Analysis or Planning
(ii) Design
(iii) Implementation
(iv) Testing
(v) Documenting
(vi) Evaluation or Deployment
(vii) Maintenance

For the moment, it is worth trying to learn the steps **(Fig. 1)** in the correct order. I usually use a silly mnemonic for this: **A Dance In The Dark Every Monday** which helps me remember **ADITDEM**:

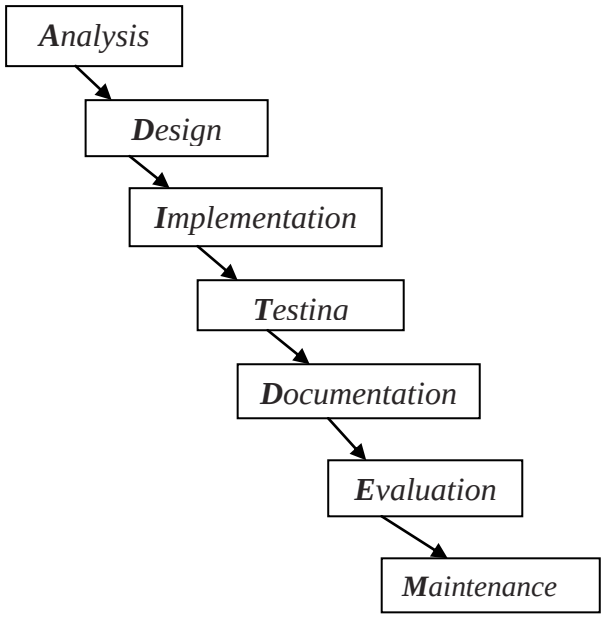

Fig. 1 Activity of the System Development Life cycle (SDLC)

We might be able to make up a better mnemonic than this one – so long as it helps you, then it's OK! Next, we will take a closer look at each of the stages.

IV. PROPOSED METHODOLOGY AND DISCUSSION

In this paper, now we discuss about the **Development Process (DP),** it focuses on how the software is to be engineered. Before we think about how software is developed, it is worth considering how any product is developed, because the process is essentially the same [10]. For example **(Fig. 2),** think about the process of developing a new model of TV.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

Fig. 2 Example of the developing a new model of TV

Stage 1: Analysis

The main purpose of the analysis stage is to be absolutely clear about what the program is supposed to do. Often, a new program will start from a rough idea. Before a new product is developed, someone within the company, probably in the marketing department, analyses what people want. They consider which products are selling well, look at what rival companies are producing, and maybe even carry out a survey to find out what people want. From this they can work out which features are required in their newest model, including its size, target price range and various technical requirements.

They use this information to produce a **specification** for the new model of TV. This state's clearly all the features that it must have [10] [11].

Stage 2: Design

The next stage is to turn the specification into a design. Designers will get to work, alone or in groups, to design various aspects of the new TV. What will it look like? How will the controls be laid out? Sketches will be drawn up and checked against the specification. Another team of designers will be planning the internal circuitry, making sure it will allow the TV to do all the things set out in the specification [12].

Two common ones are called **pseudocode** and **structure diagrams**. There are many others, but we will only consider these two. It is easy to understand these if we think about an everyday example, rather than a computer program.

Think about making tea. Here is a list of instructions for this task:

1. Get a mug out of the cupboard
2. Put a teabag in it
3. Boil the kettle
4. Pour boiling water from the kettle into the mug
5. Stir.

This is an example of **pseudocode**. It is a numbered list of instructions written in normal human language (in this case, English). It doesn't go into all the details, but it gives the main steps.

Another way of showing this is as a **structure diagram (Fig. 3).** It could look like this:

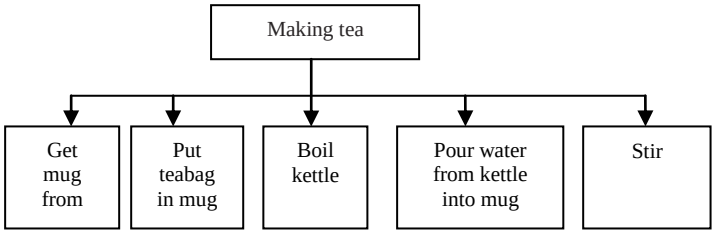

Fig. 3 Example of the structure diagram

Each instruction goes into a separate box. We read **pseudocode** from top to bottom. We read a **structure diagram** from left to right.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

Stage 3: Implementation
Once the design phase is over, engineers will get to work to actually build a prototype. Some will build the case according to the design, while others will develop the electronics to go inside. Each part will be tested on its own, then the whole thing will be assembled into a (hopefully) working TV set [13].

Stage 4: Testing
Before the new model can be put on sale, it will be thoroughly tested. A wide range of tests will be carried out. We looked at testing at the start of this section. Whether we are talking about a new TV, a new item of clothing, or a new computer program, the manufacturers will spend a great deal of time on testing. This will be carefully planned to test a wide range of conditions [14].

We can divide it up into three types of testing.

- **Testing normal conditions**
  Making sure the program does what it should do when used 'normally'. It might be tested under '**normal**' conditions. It could be put in a room at normal room temperature, and checked to see that all the controls work correctly, the display is clear, it is nice and stable, and so on.

- **Testing extreme conditions**
  Making sure the program can handle situations that are at the edge of what would be considered normal. If it passes this type of testing, it might next be tested under '**extreme**' conditions.
  For example, does it still work if the temperature is below freezing, or very hot and humid, if it used for long periods of time, or with the volume or the brightness or contrast set to their maximum values.

- **Testing exceptional conditions**
  Making sure it can handle situations or inputs that it has not been designed to cope with. Finally, it could be tested under '**exceptional**' conditions. What happens if a 2-year old picks up the remote and presses all the buttons at once? What happens if there is a power cut, or a power surge?
  If it fails any of these tests, it might be necessary to go back to the implementation (or even design) stage and do some further work, before re-testing.
  If it passes all the tests, then the new TV can go into production.

Stage 5: Documentation
However, the development isn't yet complete! Some documentation will be needed to go with the TV – a **User Manual** containing all the instructions about how to work the new TV, and probably a **Technical Manual** for repair engineers. When we buy a product, whether it is a computer program or anything else, we usually get some kind of User Guide with it (e.g. Fig. 4). This tells us how to use the product. Some software comes with a big fat book called User Guide or Manual; others come with the User Guide on a CD [15].

Fig. 4 Example of the User Guide or Manual

Stage 6: Evaluation

Once the model is in production, the company will want to evaluate it. Evaluation involves reviewing the software under various headings to see if it is of the quality required.

We will review software under these headings: Does it do what it is supposed to do? Is it easy to use? And, from the engineer's point of view, is it easy to repair? [16]

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

Stage 7: Maintenance

Stage 6 should be the end of the story, but in the real world, there needs to be stage 7 – maintenance [17].

There are different types of maintenance that might be required. These are called corrective maintenance, perfective maintenance and adaptive maintenance. We don't need to know these names until Higher level, but it is useful to think about what they mean.

Fig. 5 Example of the Maintenance improving the design

Corrective Maintenance means fixing any bugs that appear once the program is in use. Of course, these should all have been discovered during testing. However, most programs (but not the ones you will be writing) are so huge and complex that some bugs are bound to slip through unnoticed. If the bugs are serious in nature, the software company might issue a free 'patch' on its website, so that users can download the patch, and install it with the software, so fixing the bug. If it is a minor bug, they may not bother.

Perfective Maintenance is adding new features to the software. These might be suggested as a result of the evaluation stage, or they might be suggested by users. These new features will then be added to the software, and re-issued as a new version. That's why software often has version numbers. Each version results from corrective and perfective maintenance of the earlier versions. So (for e.g.), BloggProg 3.2 will be similar to BloggProg 3.1, but with bugs fixed, and some new features added.

The third type of maintenance is **Adaptive Maintenance**. This is where the software has to be changed to take account of new conditions. The most obvious example is when a new operating system comes out. Perhaps BloggProg 3.2 was designed to run under Windows 2000. When Windows XP came along, changes had to be made to BloggProg so that it would work under the new operating system.

These seven stages are an essential part of the production process.

Most software projects fail in the group of companies. In fact, they reports that more than 75% software projects are unsuccessful because they are over budget, late, missing functions or a combination of all three.

Moreover, 30% of software projects are so poorly executed that they are cancelled before completion. In my experience, software projects using modern technologies such as Java, Java 2 Enterprise Edition (J2EE), XML and Web services are no exception to this rule [1].

B. What is a Stakeholder?

A Stakeholder in the architecture of a system is an individual, team, group, organization, or classes thereof, having an interest in the realization of the system.

Most system development projects include representatives from most if not all of these stakeholders groups, although their relative importance will obviously vary from project to project.

The architect must ensure that there is adequate stakeholder representation across the board, including non-technology stakeholders (such as acquires and users) and technology-focused ones (such as developers, system administrators, and maintainers) [18].

We classify stakeholders according to their roles and concerns as in the following **table 1**:-

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November 2015

Table 1: Stakeholders Roles and Responsibility

Stakeholders Name	Roles and Responsibilities
Acquires	Oversee the procurement of the system or product.
Assessors	Oversee the system's conformance to standards and legal regulation.
Developers	Construct and deploy the system from specifications (or lead the teams that do this).
Maintainers	Manage the evolution of the system once it is operational.
Production Engineers	Design, deploy, and manage the hardware and software environments in which the system will be built, tested, and run.
Suppliers	Build and/or supply the hardware, software, or infrastructure on which the system will run.
Support Staff	Provide support to users for the product or system when it is running.
System Administrators	Run the system once it has been deployed.
Testers	Test the system to ensure that it is suitable for use.
Users	Define the systems, functionality and ultimately make use of it.

V. CONCLUSIONS AND FUTURE WORK

After completing this research, it is concluded that Testing is necessary on each phase of Software Development Life Cycle. It defines each step in the software development process describing how to develop, how to maintain and how to replace specific software. There are various types of SDLC Models for developing systems for different sizes of projects and requirements, but all of them start testing after implementation. In future, it will possible to relate new coming testing techniques with phases of SDLC, because it reduces the development time. Timing is very crucial in software development. If a delay happens in the development phase, the Market could be taken over by the competitor. Also if a 'bug' filled product is launched in a short period of time (quicker than the competitors), it may affect the reputation of the company. So, there should be a trade-off between the development time and the quality of the product. Customers don't expect a bug free product but they expect a user-friendly product that they can give a thumbs-up to.

REFERENCES

[1] Introduction to Software Engineering /Process available at https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Process
[2] Several software development approaches available at https://en.wikipedia.org/wiki/Software_development_process#Approaches
[3] Software Life Cycle Model (SLCM) available at http://www.tutorialspoint.com/software_engineering/software_development_life_cycle.htm
[4] Systems development life cycle models (SDLC) available at https://en.wikipedia.org/wiki/Systems_development_life_cycle
[5] Software Analysis available at https://en.wikipedia.org/wiki/Systems_development_life_cycle#System_analysis
[6] Software Testing available at https://en.wikipedia.org/wiki/Systems_development_life_cycle#Testing
[7] Software Evaluation available at https://en.wikipedia.org/wiki/Systems_development_life_cycle#Evaluation
[8] Rlewallen, "Software Development Life Cycle Models", 2005 http://codebeter.com.