International Journal of Computing Science and Information Technology, **2018**
ISSN: 2278-9669, April 2018 (http://ijcsit.org)

Software Engineering Development

N. Girija [1].
Sri Krishna Arts and Science College,
Coimbatore, India;
girijan16bit120@skasc.ac.in

A. Rupa Rekha [2].
Sri Krishna Arts and Science College,
Coimbatore, India,
ruparekhaa16bit145@skasc.ac.in

Received January 2018

Abstract

This paper aims to present a model of software engineering to represent its knowledge. The fundamental knowledge relating to software engineering is well described in the textbook titled "Software Engineering by Ian Sommerville that is now in its eighth edition and the white paper, Software Engineering Body Of Knowledge (SWEBOK), by the IEEE" upon which software engineering is based. This paper gives an analysis of what software engineering is, what it consists of and what it is used for software engineering is dynamic disciplines that have continuous growth in research in identifying new methods, tools and methodologies that have cause vast improvement in software development and maintenance to be more reliable and efficient. Past research critics on cost reduction, quality and flexibility have endless try to design and develop many ways to improve these sectors are still causing impacts to the software industry.

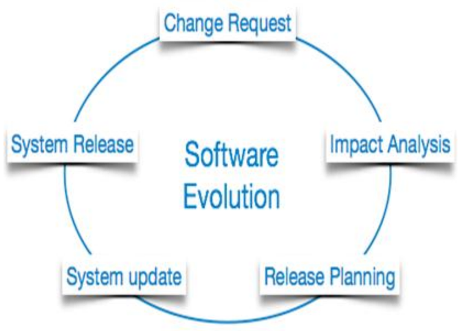
Keywords:

Introduction

Software engineering is about teams. The problems to solve are so complex or large, that a single developer cannot solve them anymore. Software engineering is also about communication. Teams do not consist only of developers, but also of testers, architects, system engineers, customer, project managers, etc. Software projects can be so large that we have to do careful planning. Implementation is no longer just writing code, but it is also following guidelines, writing documentation and also writing unit tests. But unit tests alone are not enough. The different pieces have to fit together. And we have to be able to spot problematic areas using metrics. They tell us if our code follows certain standards. Once we are finished coding, that does not mean that we are finished with the project: for large projects maintaining software can keep many people busy for a long time. Since there are so many factors influencing the success or failure of a project, we also need to learn a little about project management and its pitfalls, but especially what makes projects successful. And last but not least, a good software engineer, like any engineer, needs tools, and you need to know about them.]

SOFTWARE EVOLUTION:

The process of developing a software product using software engineering principles and methods is referred to as **software evolution.** This includes the initial development of software and its maintenance and updates, till desired software product is developed, which satisfies the expected requirements.

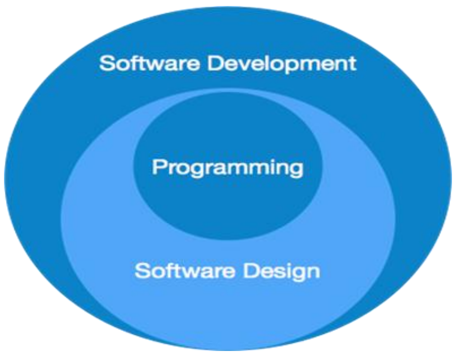


Evolution starts from the requirement gathering process. After which developers create a prototype of the intended software and show it to the users to get their feedback at the early stage of software product development. The users suggest changes, on which several consecutive updates and maintenance keep on changing too. This process changes to the original software, till the desired software is accomplished. Even after the user has desired software in hand, the advancing technology and the changing requirements force the software product to change accordingly. Re-creating software from scratch and to go one-on-one with requirement is not feasible.

The only feasible and economical solution is to update the existing software so that it matches the latest requirements.

SOFTWARE PARADIGMS:

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in the software engineering these paradigms stand. These can be combined into various categories, though each of them is contained in one another:



Programming paradigm is a subset of Software design paradigm which is further a subset of Software development paradigm.

Software Development:

This Paradigm is known as software engineering paradigms where all the engineering concepts pertaining to the development of software are applied. It includes various researches and requirement gathering which helps the software product to build. It consists of –

- Requirement gathering
- Software design
- Programming

Software Design:

This paradigm is a part of Software Development and includes –

- Design
- Maintenance
- Programming

Programming:

This paradigm is related closely to programming aspect of software development. This includes

- Coding
- Testing
- Integration.

CHARACTERISTICS OF GOOD SOFTWARE:

A software product can be judged by what it offers and how well it can be used. This software must satisfy on the following grounds:

- Operational
- Transitional
- Maintenance

Well-engineered and crafted software is expected to have the following characteristics:

Operational:

This tells us how well software works in operations. It can be measured on:

- Budget
- Usability
- Efficiency
- Correctness
- Functionality
- Dependability
- Security
- Safety

Transitional:

This aspect is important when the software is moved from one platform to another:

- Portability
- Interoperability
- Reusability
- Adaptability.

Maintenance:

This aspect briefs about how well a software has the capabilities to maintain itself in the ever-changing environment:

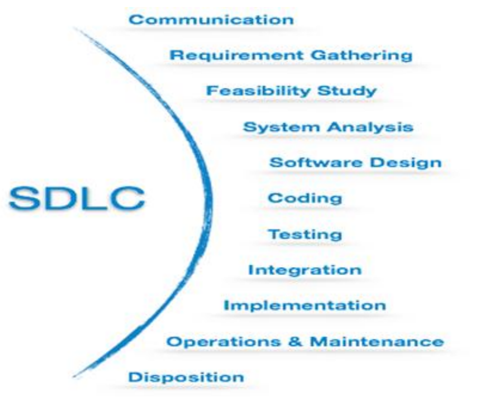
- Modularity
- Maintainability
- Flexibility
- Scalability

In short, Software engineering is a branch of computer science, which uses well-defined engineering concepts required to produce efficient, durable, scalable, in-budget and on-time software products.

Software Development Life Cycle, SDLC for short, is a well-defined, structured sequence of stages in software engineering to develop the intended software product.

SDLC ACTIVITIES:

SDLC provides a series of steps to be followed to design and develop a software product efficiently. SDLC framework includes the following steps:



Communication:

This is the first step where the user initiates the request for a desired software product. He contacts the service provider and tries to negotiate the terms. He submits his request to the service providing organization in writing.

Requirement Gathering:

This step onwards the software development team works to carry on the project. The team holds discussions with various stakeholders from problem domain and tries to bring out as much information as possible on their requirements. The requirements are contemplated and segregated into user requirements, system requirements and functional requirements. The requirements are collected using a number of practices as given -

- studying the existing or obsolete system and software,
- conducting interviews of users and developers,
- Referring to the database or collecting answers from the questionnaires.

Feasibility Study:

After requirement gathering, the team comes up with a rough plan of software process. At this step the team analyzes if a software can be made to fulfill all requirements of the user and if there is any possibility of software being no more useful. It is found out, if the project is financially, practically and technologically feasible for the organization to take up. There are many algorithms available, which help the developers to conclude the feasibility of a software project.

System Analysis:

At this step the developers decide a roadmap of their plan and try to bring up the best software model suitable for the project. System analysis includes Understanding of software product limitations, learning system related problems or changes to be done in existing systems beforehand, identifying and addressing the impact of project on organization and personnel etc. The project team analyzes the scope of the project and plans the schedule and resources accordingly.

Software Design:

Next step is to bring down whole knowledge of requirements and analysis on the desk and design the software product. The inputs from users and information gathered in requirement gathering phase are the inputs of this step. The output of this step comes in the form of two designs; logical design and physical design. Engineers produce meta-data and data dictionaries, logical diagrams, data-flow diagrams and in some cases pseudo codes.

Coding:

This step is also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently.

Testing:

An estimate says that 50% of whole software development process should be tested. Errors may ruin the software from critical level to its own removal. Software testing is done while coding by the developers and thorough testing is conducted by testing experts at various levels of code such as module testing, program testing, product testing, in-house testing and testing the product at user’s end. Early discovery of errors and their remedy is the key to reliable software.

Integration:

Software may need to be integrated with the libraries, databases and other program(s). This stage of SDLC is involved in the integration of software with outer world entities.

Implementation:

This means installing the software on user machines. At times, software needs post-installation configurations at user end. Software is tested for portability and adaptability and integration related issues are solved during implementation.

Operation and Maintenance:

This phase confirms the software operation in terms of more efficiency and less errors. If required, the users are trained on, or aided with the documentation on how to operate the software and how to keep the software operational. The software is maintained timely by updating the code according to the changes taking place in user end environment or technology. This phase may face challenges from hidden bugs and real-world unidentified problems.

Disposition:

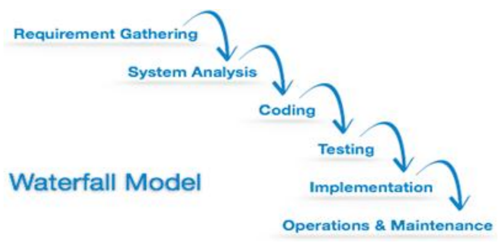
As time elapses, the software may decline on the performance front. It may go completely obsolete or may need intense upgradation. Hence a pressing need to eliminate a major portion of the system arises. This phase includes archiving data and required software components, closing down the system, planning disposition activity and terminating system at appropriate end-of-system time

SOFTWARE DEVELOPMENT PARADIGM:

The software development paradigm helps developer to select a strategy to develop the software. A software development paradigm has its own set of tools, methods and procedures, which are expressed clearly and defines software development life cycle. A few of software development paradigms or process models are defined as follows:

Waterfall Model:

Waterfall model is the simplest model of software development paradigm. It says the all the phases of SDLC will function one after another in linear manner. That is, when the first phase is finished then only the second phase will start and so on.

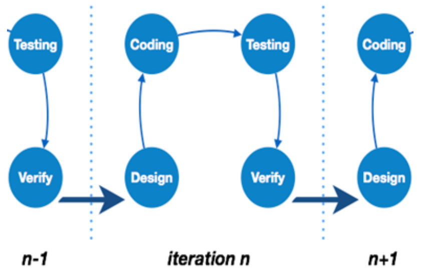


This model assumes that everything is carried out and taken place perfectly as planned in the previous stage and there is no need to think about the past issues that may arise in the next phase. This model does not work smoothly if there are some issues left at the previous step. The sequential nature of model does not allow us go back and undo or redo our actions.T his model is best suited when developers already have designed and developed similar software in the past and are aware of all its domains.

Iterative Model:

This model leads the software development process in iterations. It projects the process of development in cyclic manner repeating every step after every cycle of SDLC process.

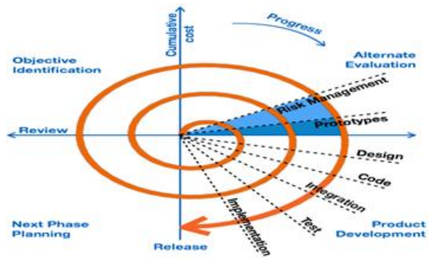
The software is first developed on very small scale and all the steps are followed which are taken into consideration. Then, on every next iteration, more features and modules are designed, coded, tested and added to the software. Every cycle produces a software, which is complete in itself and has more features and capabilities than that of the previous one.



After each iteration, the management team can do work on risk management and prepare for the next iteration. Because a cycle includes small portion of whole software process, it is easier to manage the development process but it consumes more resources.

Spiral Model:

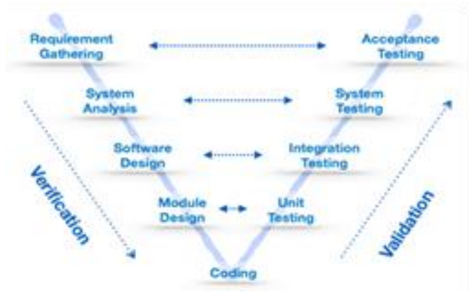
Spiral model is a combination of both, iterative model and one of the SDLC model. It can be seen as if you choose one SDLC model and combine it with cyclic process (iterative model).



This model considers risk, which often goes un-noticed by most other models. The model starts with determining objectives and constraints of the software at the start of one iteration. Next phase is of prototyping the software. This includes risk analysis. Then one standard SDLC model is used to build the software. In the fourth phase of the plan of next iteration is prepared.

V – model:

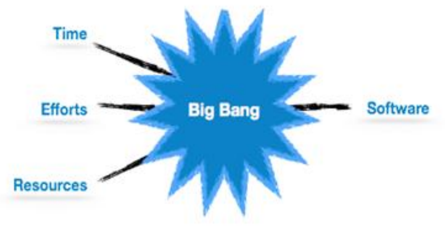
The major drawback of waterfall model is we move to the next stage only when the previous one is finished and there was no chance to go back if something is found wrong in later stages. V-Model provides means of testing of software at each stage in reverse manner.



At every stage, test plans and test cases are created to verify and validate the product according to the requirement of that stage. For example, in requirement gathering stage the test team prepares all the test cases in correspondence to the requirements. Later, when the product is developed and is ready for testing, test cases of this stage verify the software against its validity towards requirements at this stage. This makes both verification and validation go in parallel. This model is also known as verification and validation model.

Big Bang Model:

This model is the simplest model in its form. It requires little planning, lots of programming and lots of funds. This model is conceptualized around the big bang of universe. As scientists say that after big bang lots of galaxies, planets and stars evolved just as an event. Likewise, if we put together lots of programming and funds, you may achieve the best software product.



For this model, very small amount of planning is required. It does not follow any process, or at times the customer is not sure about the requirements and future needs. So the input requirements are arbitrary. This model is not suitable for large software projects but good one for learning and experimenting.

SOFTWARE TESTING LIFE CYCLE (STLC):

STLC is the testing process which is executed in systematic and planned manner. In STLC process, different activities are carried out to improve the quality of the product. Let's quickly see what all stages are involved in typical Software Testing Life Cycle (STLC).

Following steps are involved in Software_Testing Life Cycle (STLC). Each step is have its own Entry Criteria and deliverable.

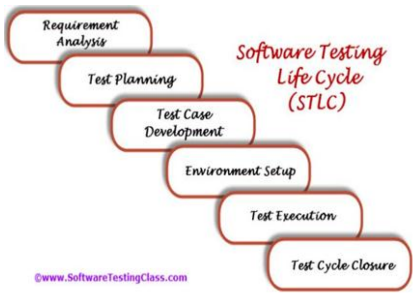
- Requirement Analysis

Test Planning

- Test Case Development
- Environment Setup
- Test Execution
- Test Cycle Closure

Ideally, the next step is based on previous step or we can say next step cannot be started unless and until previous step is completed. It is possible in the ideal situation, but practically it is not always true.

So Let's discuss what all activities and deliverable are involved in the each step in detailed.



1. REQUIREMENT ANALYSIS:

Requirement Analysis is the very first step in **Software Testing Life Cycle (STLC)**. In this step Quality Assurance (QA) team understands the requirement in terms of what we will testing & figure out the testable requirements. If any conflict, missing or not understood any requirement, then QA team follow up with the various stakeholders like Business Analyst, System Architecture, Client, Technical Manager/Lead etc to better understand the detail knowledge of requirement. From very first step QA involved in the where STLC which helps to prevent the introducing defects into Software under test. The requirements can be either Functional or Non-Functional like Performance, Security testing. Also requirement and Automation feasibility of the project can be done in this stage (if applicable).

Test Planning:

Test Planning is most important phase of **Software testing life cycle** where all testing strategy is defined.

This phase also called as Test Strategy phase. In this phase typically Test Manager (or Test Lead based on company to company) involved to determine the effort and cost estimates for entire project. This phase will be kicked off once the requirement gathering phase is completed & based on the requirement analysis, start preparing the Test Plan. The Result of Test Planning phase will be test plan or Test strategy & Testing Effort estimation documents. Once test planning phase is completed the QA team can start with test cases development activity.

Test Case Development:

The test case development activity is started once the test planning activity is finished. This is the phase of STLC where testing team write down the detailed test cases. Along with test cases testing team also prepare the test data if any required for testing. Once the test cases are ready then these test cases are reviewed by peer members or QA lead. Also the Requirement Traceability Matrix (RTM) is prepared. The Requirement Traceability Matrix is an industry-accepted format for tracking requirements where each test case is mapped with the requirement. Using this RTM we can track backward & forward traceability.

Test Environment Setup:

Setting up the test environment is vital part of the STLC. Basically test environment decides on which conditions software is tested. This is independent activity and can be started parallel with Test Case Development. In process of setting up testing environment test team is not involved in it. Based on company to company may be developer or customer creates the testing environment. Meanwhile testing team should prepare the smoke test cases to check the readiness of the test environment setup.

Test Execution:

Once the preparation of Test Case Development and Test Environment setup is completed then test execution phase can be kicked off. In this phase testing team start executing test cases based on prepared test planning & prepared test cases in the prior step. Once the test case is passed then same can be marked as Passed. If any test case is failed then corresponding defect can be reported to developer team via bug tracking system & bug can be linked for corresponding test case for further analysis. Ideally every failed test case should be associated with at least single bug. Using this linking we can get the failed test case with bug associated with it. Once the bug fixed by development team then same test case can be executed based on your test planning. If any of the test cases are blocked due to any defect then such test cases can be marked as Blocked, so we can GET the report based on how many test cases passed, failed, blocked or not run etc. Once the defects are fixed, same Failed or Blocked test cases can be executed again to retest the functionality.

Test Cycle Closure:

Call out the testing team member meeting & evaluate cycle completion criteria based on Test coverage, Quality, Cost, Time, Critical Business Objectives, and Software. Discuss what all went good, which area needs to be improve & taking the lessons from current STLC as input to upcoming test cycles, which will help to improve bottleneck in the STLC process. Test case & bug report will analyze to find out the defect distribution by type and severity. Once complete the test cycle then test closure report & Test metrics will be prepared. Test result analysis to find out the defect distribution by type and severity.

NEED OF SOFTWARE ENGINEERING:

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.

- Large software **-** It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.
- Scalability- If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.
- Cost- As hardware industry has shown its skills and huge manufacturing has lower down he price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.
- Dynamic Nature- The always growing and adapting nature of software hugely depends upon the environment in which user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.
- Quality Management- Better process of software development provides better and quality software product.

S/W ENGINEERING PRINCIPLES:

Software engineering is a layered technology. The bedrock that supports software engineering is a quality focus. The foundation for software engineering is the process layer. Software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software. Process defines a framework for a set of key process areas that must be established for effective delivery of software engineering technology. The key process areas form the basis for management control of software projects and establish the context in which technical methods are applied, work product (models, documents, data, reports, forms, etc.) are produced, milestones are established, quality is ensured, and change is properly managed. Software engineering methods provide the technical how-to's for building software. That encompass requirements analysis, design, program construction, testing, and support. Software engineering methods rely on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques. Software engineering tools provide automated or semi-automated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called computer-aided software engineering (CASE), is established. CASE combines software, hardware, and a software engineering database (a repository containing important information about analysis, design, program construction, and testing) to create a software engineering environment analogous to CAD/CAE (computer-aided design/engineering) for hardware.

CONCLUSION:

This paper takes a positive view of current progress and future challenges in software engineering. We believe the discipline has delivered and is well set to continue to deliver both practical support to software developers and the theoretical frameworks which will allow that practical support to be adopted, used and extended with confidence. It is well known that software engineering innovations take a surprisingly long time to percolate through to everyday use. Despite this lag current software engineering practice is being radically reshaped by object-oriented design methods, CASE tools with powerful code generation, testing and analysis environments, development patterns, incremental delivery based life-cycles, component models and document management environments. All of these have been formed through software engineering research. A vision of the future of software engineering suggests a setting in which developers are able to wire together distributed components and services (heterogeneous and sourced over the net) having established at an early stage, through rigorous (yet easy-to-use) formal analysis that the particular configuration will meet the requirements (both functional and non-functional). The overall process in which this takes place will have seamless tool support that extends through to change over the system or service life. Each facet of the resulting system or service will be traceable to (and from) the originating stakeholders who will be involved throughout the process.

REFERENCE:

1.https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Software_Engineering

*2.*https://www.amazon.com/Software-Engineering-9th-Ian-Sommerville/.../0137035152

3.https://www.tutorialspoint.com/software_engineering/index.htm

4.https://books.google.co.in/books?id=DuinhInx0moC&lpg=PP1&pg=PP1#v=onepage&q&f=false

5.HTTP://WWW.SOFTWARETESTINGCLASS.COM/SOFTWARE-TESTING-LIFE-CYCLE-STLC/