SOFTWARE ENGINEERING AND IT'S METHODS

Guided by Prajitha V(Ass. Proffesor, Dept. of cse , Royal college of Engineering &Technology)

Author 1:Saranya P t(Dept. of cse , Royal college of Engineering &Technology)

Author 2:Anuthama M(Dept. of cse , Royal college of Engineering & Technology)

saranyapt1996@gmail.com, anuanumohan1997@gmail.com

ABSTRACT

To develop a software product we need a systematic approach. The software engineering is the process of using an organized and regulated approach for the complete development of a software. The concept of software engineering includes a variety of methods or models. For the development of a software product a software engineer uses a specific model depending on the nature of the required software product. Each model contains a series of steps that should be accomplish to build a product. The software engineering paradigms includes Agile development model, water fall model, Spiral model, Incremental process model etc. This paper includes the concept of different methods used in the software engineering.

I. INTRODUCTION

Software engineering is an engineering discipline that's applied to the development of software in a systematic approach. It's the application of theories, methods, and tools to design a software that meets the specifications efficiently, cost-effectively, and ensuring quality. It's not only concerned with the technical process of building a software, it also includes activities to manage the project, develop tools, methods and theories that support the software production.

The effect of all previous programming experience and new ideas for writing quality programs in low cost and efficient manner have been systematically structured in to a body of knowledge. This knowledge act as the base of the principles of software engineering. software engineering methods are being widely used as a result of software crisis. The reason of

the software crisis were related to the complexity of the hardware and the process of software development. The crisis can be occurred in several ways ,

- Project running over budget
- Project running over time
- Inefficiency of software
- Low quality of software
- Software often did not meet requirements
- Difficulty in management of project and code

The software life cycle models are used to overcome the crisis involved in the development of a software product. A life cycle model that present the variety of steps or actions that required to develop a software product.

II. HISTORY

Software engineering principles have generated over the last 60 years through the results shared by the researchers and software professionals. From its beginnings in the 1960s, writing software has evolved into a profession concerned with how best to maximize the quality of software and of how to create it. Quality can refer to how maintainable software is, to its stability, speed, usability, testability, readability, size, cost, security, and number of flaws or "bugs", as well as to less measurable qualities like elegance, conciseness, and customer satisfaction, among many other attributes.

The evolution of software engineering is notable in a number of areas:

- Emergence as a profession: By the early 1980s, software engineering professionalism, to stand beside computer science and traditional engineering.
- Processes: Processes have become a big part of software engineering. They are hailed for their potential to improve software but sharply criticized for their potential to constrict programmers.
- Cost of hardware: The relative cost of software versus hardware has changed substantially over the last 50 years. When mainframes were expensive and required large support staffs, the few organizations buying them also had the resources to fund large, expensive custom software engineering projects.

Computers are now much more numerous and much more powerful, which has several effects on software. The larger market can support large projects to create commercial off the shelf software, as done by companies such as Microsoft. The cheap machines allow each programmer to have a terminal capable of fairly rapid compilation. The programs in question can use techniques such as garbage collection, which make them easier and faster for the programmer to write. On the other hand, many fewer organizations are interested in employing programmers for large custom software projects, instead using commercial off the shelf software as much as possible.

III. METHODS

A. WARERFALL MODEL

Waterfall model is the basic software development life cycle model. It is very simple but idealistic. Waterfall model divides the life cycle into a set of phases. This model considers that one phase can be started after completion of the previous phase. That is the output of one phase will be the input to the next phase. Thus the development process can be considered as a sequential flow in the waterfall. Here the phases do not overlap with each other. The different sequential phases of the classical waterfall model are shown in the below figure:

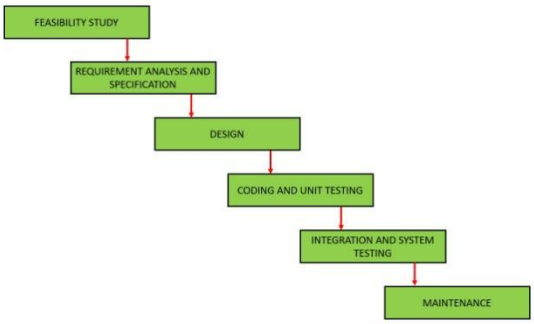

Fig III.a water fall model

- **Feasibility Study:** The main goal of this phase is to determine whether it would be financially and technically feasible to develop the software.

  The feasibility study involves understanding the problem and then determine the various possible strategies to solve the problem.

- **Requirements analysis and specification:** The aim of the requirement analysis and

specification phase is to understand the exact requirements of the customer and document them properly. This phase consists of two different activities.

- **Requirement gathering and analysis:** Firstly all the requirements regarding the software are gathered from the customer and then the gathered requirements are analyzed. The goal of the analysis part is to remove incompleteness and inconsistencies.
- **Requirement specification:** These analyzed requirements are documented in a software requirement specification (SRS) document. SRS document serves as a contract between development team and customers. Any future dispute between the customers and the developers can be settled by examining the SRS document.

- **Design**: The aim of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language.
- **Coding and Unit testing**: In coding phase software design is translated into source code using any suitable programming language. Thus each designed module is coded. The aim of the unit testing phase is to check whether each module is working properly or not.
- **Integration and System testing**: Integration of different modules are undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over a number of steps. During each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested. Finally, after all the modules have been successfully integrated and tested, the full working system is obtained and

system testing is carried out on this.

System testing consists three different kinds of testing activities as described below

- **Alpha testing:** Alpha testing is the system testing performed by the development team.
- **Beta testing:** Beta testing is the system testing performed by a friendly set of customers.
- **Acceptance testing:** After the software has been delivered, the customer performed the acceptance testing to determine whether to accept the delivered software or to reject it.

- **Maintenance:** Maintenance is the most important phase of a software life cycle. The effort spent on maintenance is the 60% of the total effort spent to develop a full software. There are basically three types of maintenance :
  - **Corrective Maintenance:** This type of maintenance is carried out to correct errors that were not discovered during the product development phase.
  - **Perfective Maintenance:** This type of maintenance is carried out to enhance the functionalities of the system based on the customer's request.
  - **Adaptive Maintenance:** Adaptive maintenance is usually required for porting the software to work in a new environment such as work on a new computer platform or with a new operating system.

Advantage

- This model is very simple and is easy to understand.
- Phases in this model are processed one at a time.
- Each stage in the model is clearly defined.
- This model has very clear and well undestood milestones.
- Process, actions and results are very well documented.

- Reinforces good habits: define-before- design, design-before-code.
- This model works well for smaller projects and projects where requirements are well understood.

Disadvantages

- **No feedback path:** In classical waterfall model evolution of a software from one phase to another phase is like a waterfall. It assumes that no error is ever committed by developers during any phases. Therefore, it does not incorporate any mechanism for error correction.
- **Difficult to accommodate change requests:** This model assumes that all the customer requirements can be completely and correctly defined at the beginning of the project, but actually customers' requirements keep on changing with time. It is difficult to accommodate any change requests after the requirements specification phase is complete.
- **No overlapping of phases:** This model recommends that new phase can start only after the completion of the previous phase. But in real projects, this can't be maintained. To increase the efficiency and reduce the cost, phases may overlap.

B. PROTOTYPING MODEL

The Prototyping Model is one of the most popularly used Software Development Life Cycle Models .This model is used when the customers do not know the exact project requirements beforehand. In this model, a prototype of the end product is first developed, tested and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product. In this process model, the system is partially implemented before or during the analysis phase thereby giving the customers an opportunity to see the product early in the life cycle. The process starts by interviewing the customers and developing the incomplete high-level paper model. This document is used to build the initial prototype supporting only the basic functionality as desired by the customer. Once the customer figures out the problems, the prototype is further refined to eliminate them. The process continues till the user approves

the prototype and finds the working model to be satisfactory.

There are 2 approaches for this model:

- **Rapid Throwaway Prototyping:** This technique offers a useful method of exploring ideas and getting customer feedback for each of them. In this method, a developed prototype need not necessarily be a part of the ultimately accepted prototype. Customer feedback helps in preventing unnecessary design faults and hence, the final prototype developed is of a better quality.
- **Evolutionary Prototyping:** In this method, the prototype developed initially is incrementally refined on the basis of customer feedback till it finally gets accepted. In comparison to Rapid Throwaway Prototyping, it offers a better approach which saves time as well as effort. This is because developing a prototype from scratch for every iteration of the process can sometimes be very frustrating for the developers.

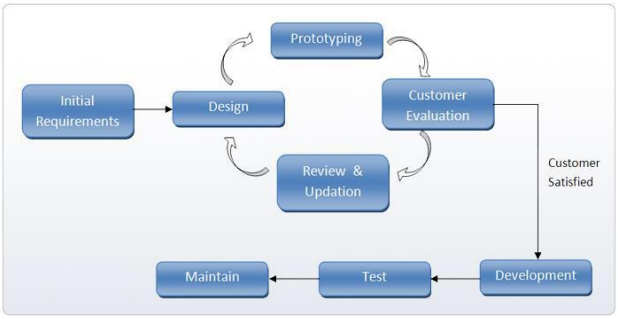

Fig III.b prototyping model

Advantages

- The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.
- New requirements can be easily accommodated as there is scope for refinement.
- Missing functionalities can be easily figured out.
- Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
- The developed prototype can be reused by the developer for more complicated projects in the future.
- Flexibility in design.

Disadvantages

- Costly w.r.t time as well as money.
- There may be too much variation in requirements each time the

prototype is evaluated by the customer.

- Poor Documentation due to continuously changing customer requirements.
- It is very difficult for the developers to accommodate all the changes demanded by the customer.
- There is uncertainty in determining the number of iterations that would be required before the prototype is finally accepted by the customer.
- After seeing an early prototype, the customers sometimes demand the actual product to be delivered soon.

C. EVOLUTIONARY MODEL

Evolutionary model suggests breaking down of work into smaller chunks, prioritizing them and then delivering those chunks to the customer one by one. The number of chunks is huge and is the number of deliveries made to the customer. The main advantage is that the customer's confidence increases as he constantly gets quantifiable goods or services from the beginning of the project to verify and validate his requirements. The model allows for changing requirements as well as all work in broken down into maintainable work chunks.

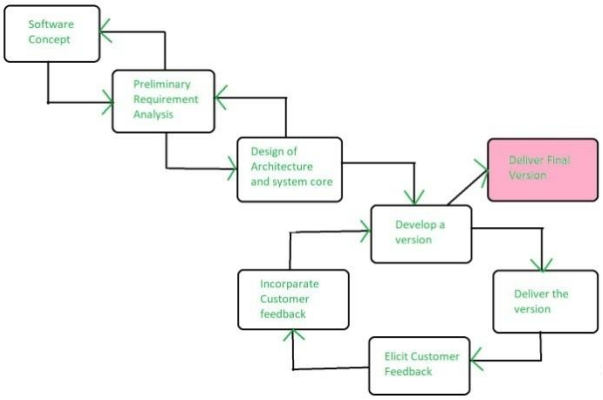

Fig III.c evolutionry model

Advantages:

- In evolutionary model, a user gets a chance to experiment partially developed system.
- It reduces the error because the core modules get tested thoroughly.

Disadvantages:

- Sometimes it is hard to divide the problem into several versions that would be acceptable to the customer which can be incrementally implemented and delivered.

D. SPIRAL MODEL

Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral

with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process. The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase. The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

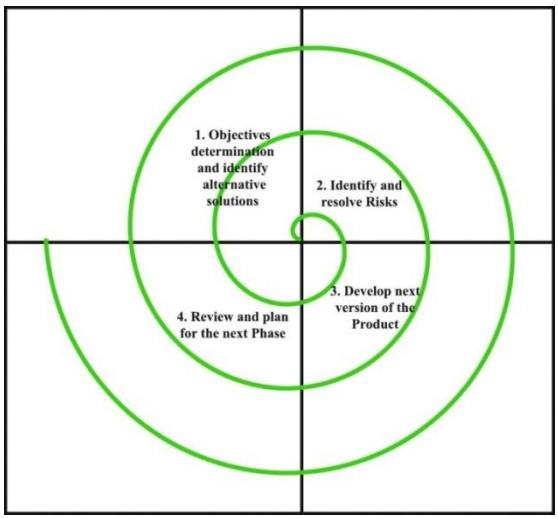

Fig III.d spiral model

Advantages

- **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
- **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
- **FlexibilityinRequirements:** Cha nge requests in the Requirements at later phase can be incorporated accurately by using this model.
- **CustomerSatisfaction:** Custome r can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

Disadvantages

- **Complex:** The Spiral Model is much more complex than other SDLC models.
- **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
- **Too much dependable on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced expertise, it is going to be a failure to develop a project using this model.
- **Difficulty in time management:** As the number of

phases is unknown at the start of the project, so time estimation is very difficult.

E. AGILE METHODOLOGY

Agile methodology is a practice that promotes continuous iteration of development and testing through out the development of project. In this design and implementations are treated as the central activities in the development process. These two phases also incorporate other activities like requirements elicitation and testing into it .In agile method iterations occurs across activities. So the requirements and design are developed together. The allocation of requirements and the design planning and development as executed in series of increments. It concentrates more on code development rather than documentation.

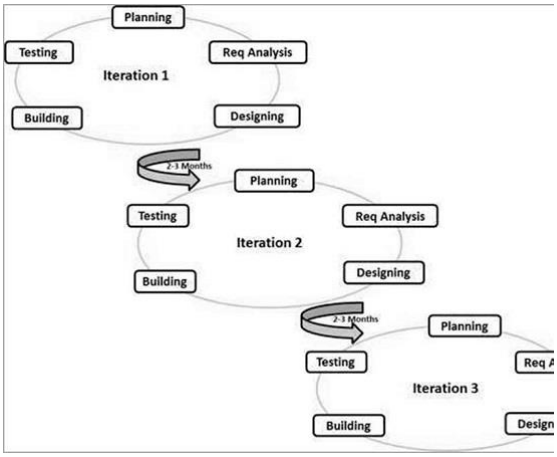

Fig III.e agile methodology

Advantages

- Deployment of product is quicker and thus helps in increasing the trust of the customer.
- Can better adapt to rapidly changing requirements and responds faster.
- Helps in getting immediate feedback which can be used to improve the software in the next increment.
- People and interactions are given a high priority rather than process and tools.

Disadvantages

- In case of large projects, it is difficult to assess the effort required at the initial stages of the development process.
- It is more code focused and produce less documentation.
- Heavily depended on the input of the customer.
- Face to face communication is harder in larger –scale organizations.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it is a difficult situation for new

programmers to adapt to the environment.

IV CONCLUSION

The concept of software engineering includes a variety of methods or models. For the development of a software product a software engineer uses a specific model depending on the nature of the required software product. Each model contains a series of steps that should be accomplish to build a product. The software engineering paradigms includes Agile development model, water fall model, Spiral model, Incremental process model etc. This paper includes the concept of different methods used in the software engineering.

V REFERENCE

[1] Rajib Mall “Fundamentals of software engineering”.
[2]RogerS.Pressman”Software Engineering”
[3] Bollinger, T. and C. McGowen, “A Critical Look at Software Capability Evaluations,”.
[4] Gilb, T., “What Is Level Six?”
[5]Hopper, M.D., "Rattling SABRE, New Ways to Compete on Information,".
[6]Paulk, M., “Capability Maturity Model for Software,”.