20/9/23

Software Engineering (SE)

Course code

AGCS07

UNIT-1

Part-I

Introduction to Software Engineering

1. The evolving role of software (S/w)
2. Changing Nature of S/w
3. Software Myths

Part-II

A Generic view of Process

1. S/w Engineering - A layered Technology
2. A process Frame work
3. The capability Maturity Model Integration (CMMI)

Part-III

Process Models

1. The Waterfall Model
2. Spiral Model
3. Agile Methodology

Software Engineering

Introduction

The term software engineering is the product of two words, software & engineering.

- The software is a collection of integrated programs.
- Slw subseeds (maintain / support) of carefully organised instructions and code written by developers on any of various particular computer languages.
- Computer programs and related documentation such as requirements, design models and user manuals.
- Engineering is the application of scientific and practical knowledge to invent, design, build, maintain and improve frame works, processes etc.

Software Engineering

S.E is an engineering branch related to the evolution of software product using well-defined scientific principles, techniques and procedures.

The result of software engineering is an effective and reliable s/w product.

Goal

Goal of the SE is "to produce high quality software at low cost."

① The Evolving role of a software

S/w takes dual role. It is both a product and a vehicle (Process) for delivering a product.

As a product : It delivers the computing potential include by computer hardware (or) by a network of computers.

As a vehicle : It is information transformer- producing, managing, acquiring, modifying, displaying (or) transmitting information that can be a simple as single bit (or) as complex as a multimedia presentation.

Software delivers the most important product of our time - information.

→ It transforms personal data

→ It manages business information to enhance competitiveness:-

→ It provides a gate way to world-wide information networks.

→ It provides the means for acquiring information

The role of computer slw has undergone significant change over a span of little more than 50 years.

Dramatic improvements in hardware performance, profound changes in computer architectures, vast increases in memory and storage capacity and a wide-variety of exhautic input & output options have more sophisticated and complex computer based system.

The lone programmer of an earlier era has been replaced by a team of slw specialists, each focusing on one part of the technology required to deliver a complex application. And yet, the same questions asked by the lone programmer are being asked when modern ~~and~~ computer based systems are built:

→ Why does it takes so long to get slw finished?

→ Why are development costs so high?

→ Why can't we find all the errors before

we give the s/w to customers?

→ Why do we continue to have difficulty in measuring progress as s/w is being developed?

These and many other questions are a manifestation of the concern about s/w and the manner in which it is developed.

Today, a huge s/w industry has become a dominant factor in the economies of the industrialised world.

(2) Changing nature of Software

The nature of slw has changed a lot over the years. Seven broad categories of computer slw continuing challenges for software engineers:

1. System slw
2. Application slw
3. Engineer / Scientific slw
4. Embedded slw
5. Product-line slw
6. Web Applications
7. Artificial Intelligence slw

1. System Software

System software is a collection of programs written to service other programs. The system slw is the interface b/w the hardware and user applications. The operating system is the best known example of system slw.

Eg: Microsoft windows, compliers, editors

2. Application Software

Application software consist of standalone programs that solve a specific business need.

- Applications in this area process business (or) technical data in a way that facilitates business operations (or) management (or) technical decision making.

- In addition to conventional data processing, application slw is used to control business functions in real-time.

  Eg: Point-of-sale transaction processing, Realtime manufacturing process-control.

3. Embedded Software

3. Engineering / scientific slw.

Engineering / scientific slw satisfies the needs of a scientific (or) engineering user to perform enterprise-specific tasks. Such software is written for specific applications using principles, techniques & formulae particular to that field.

- However, modern applications within the engineering/scientific area are moving away from conventional numerical algorithms.

  Eg: MAT LAB, AUTOCAD, PSPICE, ORCAD etc..

4. Embedded slw

Embedded slw resides within a product (or) system and is used to implement and control features and functions for the end user and for the system itself.

- It can perform
  - * Limited and esoteric functions (eg: keyboard control for microwave ovens)
  - * Provide significant function and control capability eg: Digital functions in automobile such as fuel control, dash board displays, breaking

systems etc.,

5. Product-line software

Product-line s/w is designed to provide a specific capability for use by many different customers.

. Product-line s/w can

* focus on a limited and esoteric market place

Eg: Inventory control products

(or)

* Address mass consumer markets

Eg: Word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, personal and business financial applications.

6. Web Applications

The software related to web come under this category.

Eg: CGI, HTML, JAVA, PERL, DHTML etc.,

. Web Applications are evolving into sophisticated computing environments that not only provide standalone features, computing functions and content to the end user, but also are integrated with corporate database and business applications.

7. Artificial Intelligence software

Artificial Intelligence slw makes use of non-numerical algorithms to solve complex problems that are not amenable to computation (or) straight forward analysis.

- Application within this area includes robotics, expert systems, pattern recognition (image & voice), artificial neural networks, theorem proving and game playing.

26/9/23

③ Software Myths

Slw myths - beliefs about slw and the process used to build it - can be traced to the earliest days of computing. Myths have a number of attributes that have made them insidious.

Most knowledgable professionals recognise myths for what they are misleading attributes that have caused serious problems for managers and technical people alike. However, old attitudes and habits are difficult to modify and remnants of slw myths are still believed.

i) Management Myths

Myth 1 : We already have a book i.e., full of standards and procedures for building s/w, won't that provide many people with everything they need to know ?

Reality : The book of standards may very well exist, but is it used ? Are s/w practitioners aware of its existence ? Does it reflect modern s/w engineer practice ? Is it complete ? Is it streamline to improve time and to delivery while still maintaining a focus on quality ? In many cases the answer to all of these questions is 'NO'.

Myth 2 : If we get behind schedule, we can add more programmers and catch up.

Reality : S/w development is not a mechanistic process like manufacturing.

"Adding people to a late software project makes it later". At first, the statement may seem counter-intuitive. However, as new people are added, people who were working must spend time educating the new comers, thereby reducing the amount of time spent on productive development effort. People can be added but only in a planned and well coordinated manner.

Myth 3: If I decide to outsource the slw project to a third-party, I can just relax and let that firm build it.

Reality: If an organization does not understand how to manage and control slw projects internally it will invariably struggle when it outsources slw projects.

ii) Customer Myths

Myth 1: A general statement of objectives is sufficient to begin writting programs - we can fill in the details later.

Reality: A poor upfront definition is the major cause of failed slw efforts. A formal and detailed descriptions of the information domain, function, behaviour, performance, interfaces, design constraints and validation criteria is essential. These characteristics can be determined only after thorough communication between customer and developer.

Myth 2: Project requirements continually change, but change can be easily accomidated because slw is flexible.

Reality: It is true that slw requirements change, but the impact of change varies with the time at which it is introduced. When

requirement changes are requested early (before design or code has been started) cost impact is relatively small. However, as time passes, the cost impact grows rapidly. Resources have been committed and a design framework has been established and change can cause upheaval that requires additional resources and major design modifications.

iii) Practitioner's Myths

Myth 1: Once we write the program and get it work, our job is done.

Reality: Someone once said that "The sooner you began 'writing code', the longer it will take you to get done". Industry data indicate that between 60 & 80 percent of all effort expended on slw will be expended after it is deliever to the customer for the first time.

Myth 2: Until I get the program running I have no way of assessing its quality.

Reality: One of the most effective slw quality assurance mechanisms can be applied ; from the inception of a project - the formal technical review (FTR). Slw reviews are a "Quality filter" that have been found to be more effective than testing for finding certain classes of slw errors.

Myth 3: The only deliverable work product for a successful project is the working program.

Reality: A working program is only one part of a s/w configuration, that includes many elements. Documentation provides a foundation for successful engineering and more important guidance for s/w support.

29/9/23

Part - II

Generic view of processing

① Software Engineering - A layered Technology

Software Engineering is the establishment and use of sound engineering principles in order to obtain economically; software that is reliable and works efficiently on real machines.

Bauers definition provides it with a baseline:

→ What "sound engineering principles" can be applied to computer s/w development?

→ How do we "economically" build software so that is "reliable"?

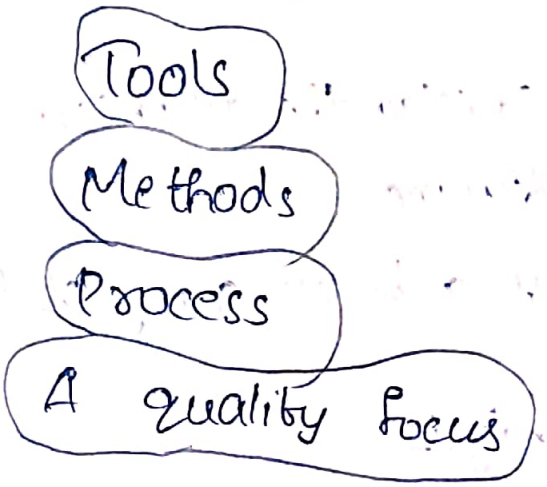
→ What is required to create computer programs that work "efficiently" on not one but many different "real machines"?

These are the qsns that continue to challenge s/w engineers

The IEEE developed a more comprehensive definition when it states <u>software engineering</u>.

* The application of a systematic, discipline, quantifiable approach to the development, operation & maintainence of slw. That is the application of engineering to slw.
* The study of approaches as in the above definition,

Software engineering is a layered technology



1. Quality Focus

Any engineering approach must rest on an organisational commitment to quality. The bedrock that supports S.E is a quality focus.

2. Process

The foundation for S.E is the process layered S.E process is the glue that holds the technology layers together and enables rational & timely development of computer slw. Process defines a framework that must be establish for effective delivery of slw engineering technology

The s/w process forms the basis for management control of s/w projects & establishes the context in which

* Technical methods are applied
* Work products are produced
* Milestones are established
* Quality is ensured and
* Change is properly managed.

3. Methods

S.E methods provide the technical "how to's" for building software

Methods encompass a broad array of tasks that include

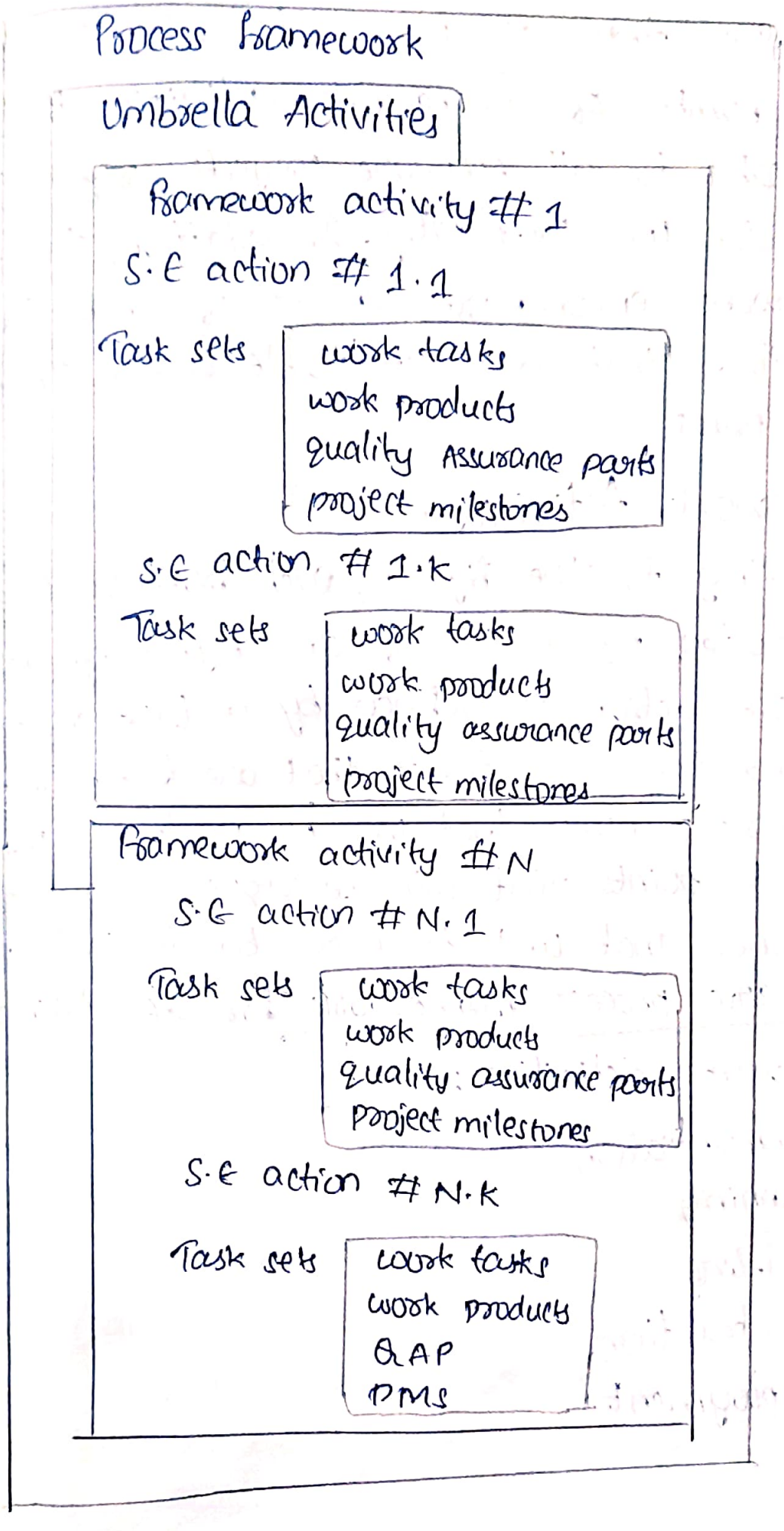
* Communication
* Requirements analysis
* Design Modeling
* Program construction
* Testing & support

4. Tools

S.E 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 tool, a system for the support of s/w development called computer aided s/w engineering is established.

(2) Process framework



4/10/23

A process framework establishes the foundation for a complete s/w process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size (or) complexity. In addition, the process framework encompasses a set of umbrella activities that are applicable across the entire s/w process

Framework Activities:

Referring to the figure, each framework activity is included by a set of 's/w engineering actions. Each S.E action is defined by a task set that identifies the work tasks that are to be completed, the work products that will be produced, the quality assurance points that will be required, and the milestones that will be used to indicate progress

A generic process framework for S.E defines 5 framework activities:

1. Communication
2. Planning
3. Modeling
4. Construction
5. Deployment

1. Communication :

This framework activity involves heavy communication & collaboration with the customer and stake holders and encompasses requirements gathering and other related activities.

2. Planning:

This activity establishes a plan for the s/w engineering work that follows. It describes the technical task to be conducted, the risks that are likely, the resources that will be required, the work products to be produced and a workschedule.

3. Modeling;

This activity encompasses the creation of models that allow the developer and customer to better understand s/w requirements and the design that will achieve those requirements.

The modeling activity is composed of two S.E actions

i) Analysis
ii) Design

i) Analysis : It encompasses a set of work tasks (Eg: requirements gathering, elaboration, negotiation, specification & validation) that lead to creation of analysis model.

ii) Design : It encompasses work tasks (data design, architectural design, interface design & component-level design) that create a design model.

4. Construction:

This activity code generation and the testing that is required to uncover the errors in the code.

5. Deployment:

The slw is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation.

These 5 generic framework activities can be used during the development of small programs, the creation of large web applications and for the engineering of large, complex computer based systems. The details of the slw process will be quite different in each case, but the framework activities remain the same.

5/10/23

Umbrella Activities:

The following are the set of umbrella activities

1. Software project tracking & control: It allows the slw team to asses progress against the project plan and take necessary action to maintain schedule

2. Risk Management: Asses risk that may effect the outcome of the project (or) the quality of the product.

3. Software Quality Assurance : It defines and conducts the activities required to ensure slw quality and also it performs actions to ensure the product's quality. This maintain time complexity, space complexity & overall quality of that product.

4. Formal Technical Reviews (FTR) : It asses S.E work products in an effort to uncover and remove errors before they are propagated to the next action (or) activity.

5. Measurement : It define & collects and process, project and product measures that assist the team in delivering slw that needs customer's needs can be used in conjunction with all other framework and umbrella activities.

6. Slw Configuration Management : It manages the effects of change throughout the slw process. Managing of configuration process is, when any change in the slw occurs. SCM has some rules & techniques because there are different linking library files, linking slw files. So it maintain the configuration of all these.

7. Reusability Management: It defines criteria for work product reuse and establishes mechanisms to achieve reusable components. Reusable work items should be backed up.

Eg: Login modules, Logout modules

8. Work product preparation & production: It encompasses the activities required to create work products such as models, documents, logs, forms & lists. It maintain the documentation of each & every activity.

(3) CMMI (Capability Maturity Model Integration)

- CMMI is a successor of CMM
- It is a more evolved model that incorporates best components of individual disciplines of CMM like slw CMM, systems engineer CMM, people CMM etc
- Since CMM is a reference model of matured practices in a specific displines, so it becomes difficult to integrate these disciplines as per the requirements.

Objectives of CMMI:

1. Fulfilling customer needs and expectations.
2. Value creation for investors / stock holders.
3. Market growth is increased.
4. Improved quality of products and services.

5. Enhanced reputation in Industry.

CMMI Representation:

A representation allows an organization to pursue a different set of improvement objectives. There are two representations for CMMI.

1. Staged Representation
2. Continuous Representation

1. Staged Representation:

- Uses a pre-defined set of process areas to define improvement path.
- Provides a sequence of improvements, where each part in the sequence serves as a foundation for the next.
- An improved path is defined by maturity level.
- Maturity level describes the maturity of processes in organization.
- Staged CMMI representation allows comparison b/w different organisations for multiple maturity levels.

2. Continuous Representation:

- Allows selection of specific process area.
- Uses capability levels that measures improvement of individual process area.
- Continuous CMMI Representation allows comparison b/w different organizations on a process-area

by process-area basis.

- Allows organisations to select processes which require more improvement.
- In this representation, order of improvement of various processes can be selected which allows the organisations to meet their objectives and eliminate risks.

CMMI Model - Maturity levels:

In CMMI with staged representation, there are five maturity levels.

i) Maturity level 1 : Initial

→ Processes are poorly managed or controlled.
→ Unpredictable outcomes of processes involved.
→ Adhoc and Chaotic approach used.
→ No KPA (Key Process Area) defined.
→ Lowest quality and highest risk.

ii) Maturity Level 2 : Managed

→ Requirements are managed.
→ Processes are planned and controlled.
→ Projects are managed and implemented according to their documented plans.
→ Risk involved in this level is lower than initial level, but still exists.
→ Quality is better than initial level.

iii) Maturity level 3: Defined

→ Processes are well characterised and described using standards proper procedures and methods, tools, etc...

→ Medium quality and medium risk involved.

→ Focus is process standardization.

iv) Maturity Level 4: Quantitatively Managed

→ Quantitative objects for process performance and quality are set.

→ Quantitative objectives are based on customer requirements, organization needs, etc.

→ Process performance measures are analysed quantitatively.

→ Higher quality of processes is achieved.

→ Lower risk.

v) Maturity Level 5: Optimizing

→ Continuous improvement in processes and their performance.

→ Improvement has to be both incremental and innovative.

→ Highest quality of processes.

→ Lowest risk in processes and their performance.

CMMI Model - capability Levels:

A capability level includes relevant specific and generic practices for a specific process area that can improve the organization's processes associated with that process area.

→ For CMMI models with continuous representation, there are six capability levels.

ⓐ Capability level 0: Incomplete

* Incomplete process - partially or not performed.
* One or more specific goals of process area are not met.
* No generic goals are specified for this level.
* This capability level is same as maturity level 1.

ⓑ Capability level 1: Performed

* Process performance may not be stable.
* Objectives of quality, cost & schedule may not be met.
* A capability level 1 process is expected to perform all specific and generic practices for this level.
* Only a start-step for process improvement.

ⓒ Capability level 2: Managed

* Process is planned, monitored and controlled.
* Managing the process by ensuring that objectives are achieved.

* Objectives are both model and other including cost, quality, schedule.
* Actively managing processing with the help of metrics.

(d) Capability level 3: Defined
* A defined process is managed and meets the organization's set of guidelines and standards.
* Focus is process standardization.

(e) Capability level 4: Quantitatively Managed
* Process is controlled using statistical & quantitative techniques.
* Process performance and quality is understood in statistical terms and metrics.
* Quantitative objectives for process quality and performance are established.

(f) Capability level 5: Optimizing
* Focus on continually improving process performance.
* Performance is improved in both ways - incremental and innovation.
* Emphasizes on studying the performance results across the organisation to ensure that common causes or issues are identified and fixed.

12/10/23 (11) Process Models

Introduction:-

In S.E, a slw process model is the mechanism of building slw development work into distinct phases to improve design, product, management & project management it is also known as slw development life cycle.

It establishes the foundation for a complete slw process by identifying a small no. of framework activities. It includes a set of umbrella activities that are applicable across the entire slw process. Each framework activity is populated by a set of S.E actions. A generic process framework for S.E encompasses five activities.

(i) Communication
(ii) Planning
(iii) Modeling
(iv) Construction
(v) Deployment

→ Process modeling is the graphical representation of business process (or) work flows. Slw process is the set of activities & associated outcome that produce a slw product, which are common to all slw processes.

Waterfall Model :- (WFM)

The WFM, sometimes called the classic lifecycle, suggests a systematic sequential approach to slw development that begins with customer specification of requirements & progress through planning, modeling, construction & deployment.

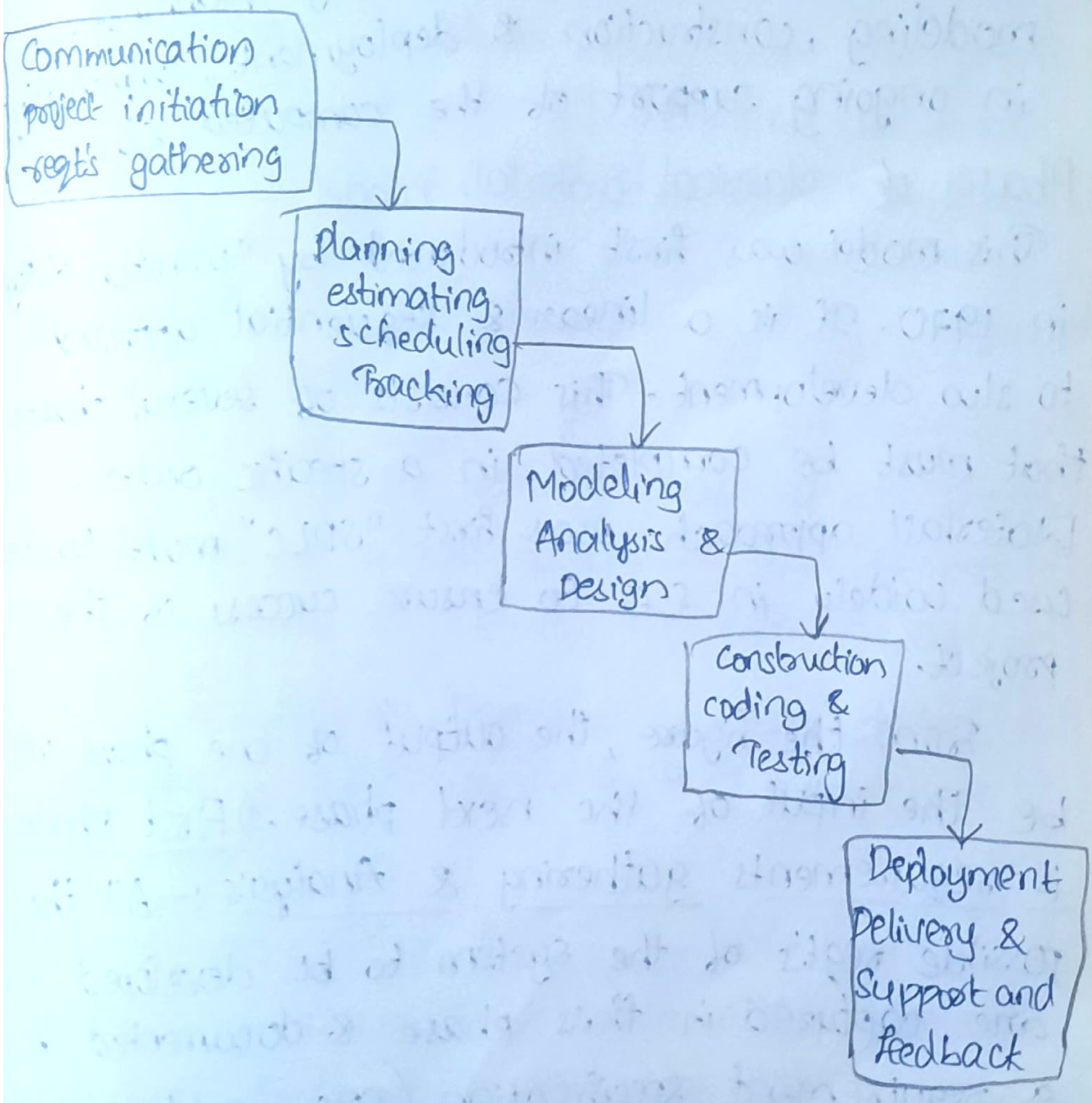


fig: The waterfall Model.

The Waterfall Model (WFM):-

It is often used for large scale projects with long timelines, where there is a little room (scope) for errors & project stakeholders need to have high level of confidence in the outcome. This model begins with customer specification of requirements & progressive through planning, modeling, construction & deployment, culminating in ongoing support of the computed s/w.

Phases of classical waterfall model:-

This model was first introduced by "winston W. Royce" in 1970. It is a linear & sequential approach to s/w development. This consists of several phases that must be completed in a specific order. Waterfall approach was first "SDLC" model to be used widely in S.E to ensure success of the project.

From the figure, the output of one phase will be the input of the next phase. 1) Phase-1 First phase is requirements gathering & Analysis:- All the possible reqts of the system to be developed are captured in this phase & documented in a requirement specification (SRS - Software Requirements Specifications).

phase-2
2) System Design :- The requirements specifications from 1st phase are studied & the system design is prepared. This system design helps in specifying hardware & system requirements & helps in defining the overall system architecture.

phase-3
3) Implementation :- With inputs from the system design, the system is 1st developed in small programs called units, which are integrated in the next phase. Each unit is developed & tested for its functionality, which is referred to as "unit testing".

phase-4
4) Testing :- Once the testing phase comes, the s/w is tested as a whole to ensure that it meets the requirements & it is free from defects.

phase-5
5) Deployment :- Once the s/w has been tested & approved, it is deployed to the production environment.

phase-6
6) Maintenance :- The final phase which involves fixing any issues that arise after the s/w has been deployed & ensuring that it continues to meet the requirements over the time.

Advantages of WFM :-

Easy to understand : It is very simple and easy to understand.

Individual processing : Phases in this model are processed one at a time.

Properly defined : Each stage in this model is clearly defined.

Clear milestone : It has very clear and well-understood milestones.

Properly documented : Process, actions & results are well documented.

Reinforces good habits : This model reinforces good habits like define before design and design before code.

Working :

It works well for smaller projects & project where requirements are well understood.

This waterfall model has several benefits as it helps projects keep a well defined predictable project under budget.

Disadvantages of WFM :-

Because of some major drawbacks of this model. We can't use it in real projects, but we use other software development lifecycle

models which are based on the classical waterfall model.

No feedback path : This model assumes that no error is ever committed by developers during any phase. Therefore it doesn't incorporate any mechanism for error correction.

Difficult to accommodate change requests : When customer 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 : In this model new phase can start only after the completion of previous phase. But in real project this can't be maintained. To increase efficiency & reduce cost phases may overlap.

Limited flexibility : Which is not well-suited for projects with changing or uncertain requirements. Once a phase has been completed it is difficult to make changes or go back to a previous phase.

Limited stake-holder involvement : The stakeholders are typically involved in the early phases of the project requirements gathering and analysis but may not be involved in the later

phases (implementation, testing, deployment)

Lengthy development cycle: This model can result in lengthy development cycle as each phase may be completed before moving on to the next. This can result in delays & increased costs if requirement changes or new issues arise.

Not suitable for complex projects: This model can make it difficult to manage multiple dependencies and inter-related components.

Applications of WFM :-

Large scale slw development projects: This model ensures that, this project is completed on time and within budget.

Projects with well-defined Requirements: As the sequential nature of the model requires a clear understanding of the projects objectives & scope

Projects with stable requirements: This model is well-suited for projects with stable requirements as the linear nature of the model does not allow for changes to be made once a phase has been completed.

22/11/23

(2) Spiral Model

It comes under iterative process model it is an evaluationary model. It has 4 stages. It is an endless loop, it never ends, it contains repeatative activities in this, where risks are predictable then we can use spiral model. It starts at Q_1 and ends at Q_4. In this the customer evaluation will be done. If the customer is not satisfied then one more iteration will go up. Like that again this spiral with a loop continues until & unless the customer is satisfied.

In first iteration it contains the core product and in next iteration it contains the subsequent phases and at last the product will be completed. The first phase is one round i.e., 360°.

The radius of the spiral increases, cost also increases as the radius keeps on increasing, the cost also increases. Everything will increase as the radius of the spiral keeps on increasing.

Angle will indicate the progress. For example, the angle is 180° - half of the task is done. The angle will represent the progress of the task spiral is looped until customer is satisfied. This is suitable for large projects. And also also is flexible, complex & it takes time.

Define objective:

Requirements gathering & analysis.

Identify & resolve risks:

In this identify all the risks and try to resolve those risks.

Developing next version of slw:

Nothing but interacting with the customers and taking the feedback from them, if there are any changes then those changes will be implemented.

Review & plan for next phase:

Next phase is nothing but the second iteration all those decisions will be made in this step.

If 1 module is completed then deliver it to the customer and it can be released to the customer when that module is stable, then only we take it to the next module, we test it then only we perform the integration between the 2nd module & 1st module. Then after we test everything we can release it to customer. Spiral model is also called as iterative model and also called incremental model.

Advantages:

- Requirement changes are allowed after every cycle.
- It is also called as controlled model.
- Testing is done for every cycle, before going to the next cycle.

- Customer will get to use the slw for every module.
- It is a controlled model to adapt since here we test the module & once it is stable then only we can go for the next module.

Disadvantages

- Requirement changes are not allowed in between the cycle.

```
#include<stdio.h>
#include<stdlib.h>
struct Node{
        int data;
        struct Node* link;

};
struct Node* start = Null;
struct Node* createNode(int data){
        struct Node * newNode =(struct Node*)mallo
                                        (sizeof(struct
                                        Node
        if(!newNode){
          printf("memory allocation failed.\n");
          exit(1);
        }
        newNode->data = data;
        newNode-> link = NULL;
        return newNode;
    }
void insertFront(int data) {
      Struct Node* newNode =(createNode(data);
       newNode-> link = start;
       start-= NewNode;
}
void insertRear (int data) {
      struct Node* newNode = createNode(data).
             if(!start)start =NewNode;
```

```
else {
    struct Node* current = start;
    while (current->link) current = current->link;
    current->link = newNode;
  }
}
```

AGILE MODEL

Topics covered

1 About Agile Model

2. Traditional vs Agile model working with example

3. When to use the Agile model

4. Agile principles.

5. Advantages of Agile Model

6. Disadvantages of Agile Model.

About Agile Model

→ Mostly used model in today's digital era.

→ Agile means "The ability to respond to changes from requirements technology and people".

→ It is an incremental and iterative process of software development.

2. working with example:

→ Divides requirements into multiple iterations and provides specific functionality for the release.

→Delivers multiple software requirements

→Each iteration lasts for two to three weeks.

→Direct Collaboration with customers.

→ Rapid project development.

Traditional vs Agile Model working with Example

For example:

→ Instagram social Application:

Requirements are.

1. Follow-unfollow option

2. Edit profile

3. search

4. Messaging

5. Post photos

6. upload story

7. To make reels.

8. Go Live

→ Agile model divides the complete requirements into the multiple iterations.

→ And they develop the product as per the priority of the requirements

→ In the below paragraphs, the time required for development of the above example is taken both for Waterfall model and Agile model.

Suppose in Waterfall Model, it takes two months for requirements gathering and analysis phase. After that it take 1/1/2 months for designing pupose and after that it takes 4 months approximately for coding purpose and 1 1/2 month for testing pupose and at the end, if any changes are required by the customer, let us assume the time to be for one month. We require 8 to 9 months for development purpose only.

and

Now in Agile Model in first iteration they take 3 to 4 weeks for development. After developing first iteration, they move to second iteration for 3 to 4 week and again move to next 3 to 4 weeks. For development purpose they require only 3 months. This is the reason why Agile model is in use in each and every software Development now a days. Agile model requires minimum time for development with greater accuracy and greater quality of the product.

WHEN TO USE THE AGILE MODEL

1. When project's size is large
2. When frequent changes are required. A pro
3. When highly qualified and experienced team is available
4. When a customer is ready to have a meeting with a software team all the time after each and every iteration.
5. Projects with flexible timelines and budget.

AGILE PRINCIPLES (12 PRINCIPLES)

1. Highest priority is to satisfy the customers to early and continue delivery of software
2. Being flexible about changing requirements at any point of development.
3. Working on frequent and short deliveries like couple of weeks or months with preference
4. Transparency between business people and developers and requires them to work together.
5. By providing a better productive environment and providing them with all the support, motivation. It leads to better productivity.
6. Face to face communication as the most effective way to communicate between customer and development team.
7. Continuous attention towards effective designing and Technical Excellence through following optimal code standard.
8. It promotes sustainable development because of the work of developers, users and sponsors as all of them work together.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity is the art of maximum result and less hard work by removing unnecessary tasks and prioritizing activities.
11. The best architectures, requirements and designs emerge from self-organizing and experience teams.
12. For developing effective software, regular analysis and work on improving the overall delivery or the development process.

Advantages:

1. Supports customer involvement and customer satisfaction.
2. Strong Communication of the software team with the customer.
3. Little planning required.
4. Efficient design and fulfils the business requirement.
5. Anytime changes are acceptable.
6. Provides a very realistic approach to software development.
7. Updated versions of functioning software are released every week.
8. It reduces total development time

Disadvantages:

1. Due to lack of proper documentation, once the project completes and the developers are allotted to another project, maintenance of the finished project can become difficult.
2. Depends heavily on customer interaction, so if the customer is not clear, team can be driven in the wrong direction.