Software Engineering and its models

Unit 1

Red : Indicates important

At the end of this unit we learn:

- What is software engineering ?
- Difference between software and other engineering products
- Different software development models
- Capability Maturity Model (CMM)
- Key Process Areas (KPA) of different levels of CMM
- Software Development Life Cycle (SDLC)

Software Engineering

- It is the application of systematic, disciplined and quantifiable approach to the development, operation and maintenance of a software.
- Programming includes only the coding part whereas software engineering includes coding, testing, cost estimation, time estimation, designing, maintenance etc....

Difference between software and other engineering products

Software	Engineering Products
Once developed can be changed.	Once developed cannot be changed. To accommodate change the product must be redesigned and remanufactured.
Not visible i.e. Software is developed and not manufactured.	Visible
Doesn't fail in the traditional sense (can be run any number of times without wear and tear).	Has wear and tear in operation.
Even though the design is good, software can never be 100% error free.	Can be perfectly designed.

Difference between software and other engineering products

Software	Engineering Products
Testing is identification of test cases in which software fails.	Full load testing. **Load testing** is performed to determine a system's behavior under both normal and anticipated peak **load** conditions.
Reused	Mostly can’t be reused.
Not vulnerable to external factors.	Vulnerable to external factors.

Software Development Models

- Build and Fix
- Waterfall
- Iterative Enhancement
- Prototyping
- Spiral
- RAD approach

Build and Fix Model

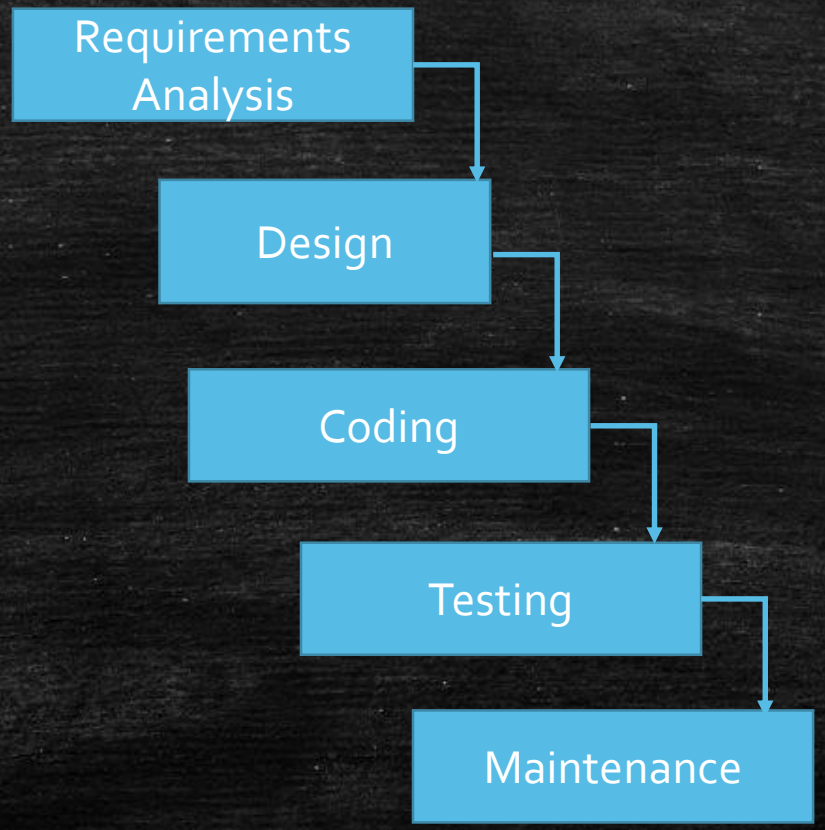
- Simple, two phase model
  - First phase: develop code
  - Second phase: fix code
- Once the code is developed, code is fixed according to customers needs and delivered to the customer.

Waterfall

- Simplest , oldest and widely used model.
- Each phase of the lifecycle is completed before the next starts.
- First engineering approach.
- Systematic and sequential.
- Requirements must be known before hand.
- No prototype is developed.

Phases of waterfall model

- Phases :
  1. Requirements analysis
  2. Design
  3. Coding
  4. Testing
  5. Maintenance



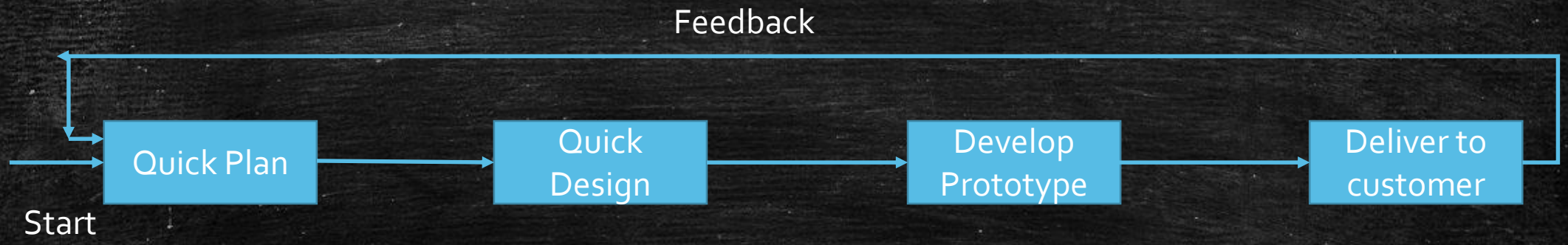
Iterative Enhancement Model

- Developed for removal of shortcomings of waterfall model
- Construction and delivery is done in iterative mode
- In first iteration, a less capable product is constructed and delivered. It satisfies only a subset of the requirements. Each iteration has all the phases of the waterfall model. Product is delivered release by release.
- Useful when less man power is available for development and when release dates are tight.
- Useful for in-house product development i.e.. When user has something to start with.
- Disadvantages:
  - Cost estimation is difficult
  - Iteration may never end, user has to endlessly wait for the final product.

Prototyping Model

- A working model of the software is developed initially.
- Prototype is a software with less functional capabilities and less reliability and does not undergo rigorous testing.
- Developing a prototype initially overcomes the disadvantage of waterfall model where serious errors and problems are reported after software development.
- Working prototype is given to the user for operation. The user operates and gives feedback. Using the feedback, developer refines, adds and prepares final specification document. Once the prototype is final, the actual software is developed using waterfall model.
- Features:
  - Helps in determining requirements more deeply.
  - At the time of actual product development, customer feedback is available.
  - Any type of risk is considered at initial phase.

Prototyping Model



Spiral Model

- Aka Boehm's model
- A risk driven process model(success of a project highly depends on the risk analysis)
- A combination of strengths of several process model
- Spiral process
- Activities:
  - Objective: set objective for the particular phase.
  - Risk analysis: identify risk, analyze and take steps.
  - Development: based on the risks identified, an SDLC model is selected and followed.
  - Planning: all work done till that time is reviewed and decided whether to go through the loop of spiral again or not.

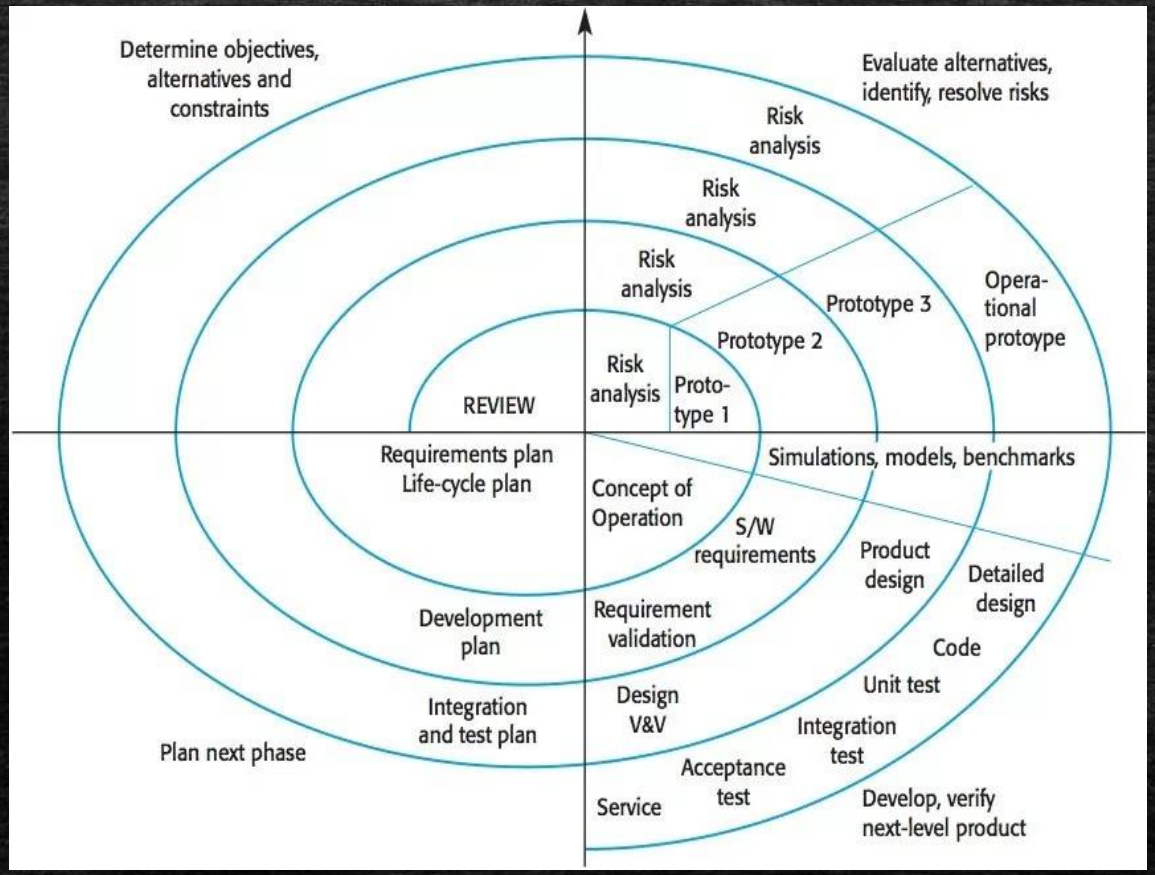
- Radical dimension represents cost estimation
- Angular dimension represents progress in completing the life cycle

- When to use?
  - For projects where risk analysis and cost estimation is important
  - For medium to high risk project.
  - When requirements are unclear and complex
  - Large projects
  - Significant changes are required

Pros and Cons of Spiral model

- Pros:
  - Development is faster.
  - Proper risk analysis is done.
  - A prototype is developed.
  - Changes can be accommodated easily
  - Useful for large projects
- Cons:
  - Not suitable for small projects
  - Risk analysis requires high expertise
  - Management is complex
  - Costly to use
  - Spiral may go indefinitely

Spiral Model



RAD Approach

- Stands for Rapid Application Development
- It's a quick approach for software development
- Software is constructed on component basis
- Overall speed of software development increases
- Fully functional system is ready in very short time
- Program components can be reused
- Follows a modular approach for development
- Problem: model may not work when technical risks are high

Capability Maturity Model

- Model is based upon capabilities of software.
- It defines the quality of the product.
- Developed by SEI(Software Engineering Institute)
- Gives emphasis to the techniques to improve software quality and process maturity.
- CMM has five maturity levels:
  - Initial
  - Repeatable
  - Defined
  - Managed
  - Optimizing

Initial

- In this, software is developed on an ad hoc basis
- No strategic approach is used for software development
- Success of project depends on the skills of the team members.
- Time and cost are not critical issues(because no particular engineering approach is followed).
- Software process is unpredictable(This is because when the developing team changes software will also change)
- Not repeatable because processes are not sufficiently defined and documented.
- Testing is also very simple.
- No key process area at this level
- Vast majority of software organizations are level 1 organizations.

Repeatable

- Organization satisfies the requirements of level 1.
- Basic project management policies and related procedures are established.
- Organization learn with experience of previous projects and reuse the successful practices. Effective process can be categorized as practiced, documented, implemented and trained.
- Manager provides quick solution to problems encountered in software development and corrective action is taken immediately. Hence process of development is much disciplined.
- Without measurement cost, functionality and schedules estimates are performed.
- Organizations have installed basic management controls.

Defined

- Organization satisfies all the requirements of level2.
- Software development processes are well-defined, managed and documented.
- Training is given to the members to gain required knowledge.
- Standard practices are simply tailored to create new projects.

Managed

- Organization satisfies all the requirements of level3
- At this level, quantitative standards are set for software products and processes.
- Project analysis is done at integrated organizational level and collective database is created.
- Performance is measured in integrated organization level.
- Software development is performed with well-defined instruments.
- Organizations capability is predictable, because projects control their processes and products to ensure their performance within qualitatively specified limits.
- Quality of software is high.

Optimizing

- Organization satisfies all the requirements of level 4.
- This is the last level.
- Organizations in this level are considered almost perfect.
- processes are constantly being improved through monitoring feedback from lower level.
- Organization analyses its weakness and takes required corrective measures to proactively prevent the errors.
- Based on cost effective analysis of new technologies, the organization changes the software development processes.

Key Process Area

- KPA is an indicative measurement of goodness of software engineering functions like project planning, requirements management etc....
- KPA consists of the following parameters:
  1. Goals: objectives to be achieved.
  2. Commitments: Requirements that the organization should meet to ensure the claimed quality of the product.
  3. Abilities: Capabilities an organization has.
  4. Activities: Specific tasks required to achieve KPA function.
  5. Methods for varying implementation: It explains how the KPAs can be verified.

KPA and Maturity level

Level	KPAs	Description
Level 1	No KPA	
Level 2	• Software Project Planning	Gives plan for software management.
	• Software Project tracking and oversight	Establish adequate visibility into actual process to enable the organization to take immediate step if it deviates from plans.
	• Requirements management	Requirements are well specified to develop a contract between developer and customer.

KPA and Maturity level

Level	KPAs	Description
Level 2	• Software subcontract management • Software quality assurance(SQA) • Software Configuration Management(SCM)	Select qualified software contractors and manage them effectively. To assure the quality of the product developed. Establish and maintain integrity through out the lifecycle of project.

KPA and Maturity level

Level	KPAs	Description
Level 3	• Organization Process Focus(OPF) • Training Program(TP) • Organization Process Definition(OPD)	Coordinate and integrate processes across all projects. Imparts training to develop the skills and knowledge of the staffs. Develop and maintain a usable set of software process assets that improve cumulative long-term benefits of the organization by improving process performance.

KPA and Maturity level

Level	KPAs	Description
	• Integrated Software Management(ISM) • Software Product Engineering(SPE) • Intergroup Coordination(IC)	Software management and software engineering activities are defined and tailored from organizational standard processes based on the requirements. Technical activities of the project are well-defined to produce correct, consistent product effectively and efficiently. Software engineering groups actively participate with other groups.

KPA and Maturity level

Level	KPAs	Description
Level 3	• Peer Review	They remove defects from software engineering products. Gives better understanding of the product.
Level 4	• Quantitative Process Management(QP)	It defines quantitative standards for software process.
	• Software Quality Management(SQM)	It develops quantitative understanding of the software quality and achieves specific quality goals.

KPA and Maturity level

Level	KPAs	Description
Level 5	• Defect Prevention (DP)	It discovers the causes of defects and devise the techniques which can prevent them from recurring.
	• Technology Change Management(TCM)	Continuously upgrades itself according to new tools, methods and processes.
	• Process Change Management(PCM)	It continuously improves the software processes used in the organization to improve software quality, increase productivity and decrease cycle time for product development.

Software Development Life Cycle

- The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process.
- Aka Software Process Technology
- Phases of SDLC are:
  1. Requirements Analysis
  2. Design
  3. Coding
  4. Software testing
  5. Maintenance

Requirements Analysis

- It describes "what the system does".
- Objectives which are to be achieved in software process development are requirements.
- In this phase, the requirements are properly defined and noted down.
- Output of this phase is SRS.
- Requirements Analysis may be defined as:
  1. The process of studying user's needs to arrive at a definition of system hardware and software requirements.
  2. The process of studying and refining system hardware and software requirements.

Design

- In this phase, a logical system is built which satisfies the requirements.
- It transforms customers requirements into a logically working system.
- Input of this phase is SRS
- Output is SDD (Software Design Document)
- Design is performed in the following two steps:
  1. Primary Design phase
  2. Secondary Design Phase

Steps in Design Phase

- Primary Design Phase
  - System is designed at block level.
  - Blocks are created based on the analysis in problem identification phase.
  - Different blocks are created for different functions.
- Secondary Design Phase
  - In this detailed design of every program is performed.

General tasks in the design process

1. Design various blocks for overall software processes.
2. Design smaller, workable and compact modules in each block.
3. Design various database structures.
4. Specify details of programs to achieve desired functionality.
5. Design the form of input and outputs of the system.
6. Perform documentation of the system.
7. System reviews.

Points to remember while designing

- It should completely and correctly describe the system.
- It should precisely describe the system. It should be understandable to the software developer.
- It should be done at the right level.
- It should be maintainable.
- Practicality: It ensures that the system is stable and can be operated by a person with less knowledge.
- Efficiency: This involves accuracy, timeliness and comprehensiveness of system input.
- Flexibility: The system must be modifiable depending upon the changing needs of the user.
- Security: This is an important aspect. It must take care of areas like security of data, hardware reliability, fall back procedures and provision for fraud detection.

Coding

- Input is SDD
- In this phase, the design document is coded according to the module specification.
- Transforms SDD into high level language code.
- Good coding standards must be followed. It makes the code more understandable.
- Coding standards gives guidelines about:
  1. Name of the module
  2. Internal and external documentation of source code
  3. Modification history
  4. Uniform appearance of codes

Testing

- Running of system on manually created inputs with the intention of finding errors.
- While developing the code, developers carry out testing known as debugging.
- Testing is meant to find the existence of defects while debugging stands for locating the place of errors and correcting the errors during the process of testing.
- Reliability of software depends on testing.
- Guidelines for testing:
  1. Test the modules thoroughly.
  2. Test the modules deliberately by passing wrong data.
  3. Create data for conditional statements.
  4. Test for locking by invoking multiple concurrent processes.

- Objectives to be kept in mind while performing testing:
  1. It should be done with the intention of finding the errors.
  2. Good test cases should be designed which have a probability of finding errors.
  3. A success test is one that uncovers undiscovered error.
- Principles of testing:
  1. All tests should be performed according to user requirements.
  2. Planning of tests should be done long before testing.
  3. Testing should start from small tests and then proceed to large tests.

Testing Strategies

1. Code Testing: It examines the logic of the system. In this, the analyst develops test cases or every instruction in the code. All the paths in the program are tested. It does not indicate if the code is according to requirements or not.

2. Specification Testing: In this, testing with specific cases is performed. Test cases are developed for each condition and combination of condition and submitted for processing.

3. Testing Techniques: Chapter 4 black box and white box testing.

Maintenance

- Software maintenance is done :
  1. To rectify the errors which are encountered during the operation of software.
  2. To change the program function to interface with new hardware or software.
  3. To change the program according to increased requirements.
- 3 categories of maintenance:
  1. Corrective maintenance
  2. Adaptive maintenance
  3. Perfective maintenance

Problems associated with maintenance

- Very cumbersome to analyze and understand the code written by somebody.
- No standards for maintenance have been developed and the area is an unexplored area.
- Few tools and techniques are available for maintenance.
- Its an necessary evil and delegated to junior programmers.