SOFTWARE ENGINEERING

LECTURE NOTES

* * *

PREPARED BY

Dr. J. Malla Reddy
Professor, Dept. of CSE

* * *

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MAHAVEER INSTITUTE OF SCIENCE & TECHNOLOGY
(Affiliated to J.N.T. University, Hyderabad)
Keshavgiri(post), Bandlaguda, Hyderabad – 500 005

APRIL, 2024

SOFTWARE ENGINEERING

Course : B.Tech [CSE] II YR / II SEM

SYLLABUS :

- **UNIT – I** : Introduction to Software Engineering, A Generic View of process, Process Models.
- **UNIT- II** : Software Requirements, Requirement Engineering Process, System Models.
  UNIT –III : Design Engineering, Creating a Architectural Design.
- **UNIT-IV** : Testing Strategies, Product Metrics.
- **UNIT –V** : Metrics for Process & Products, Risk Management.

Prescribed Books :

- **1. Software Engineering, A practitioner's Approach** - Roger S. Pressman, 6th edition, Mc Graw Hill ,International Edition. Prescribed [UNITS – 1,3,4,5]
- **2. Software Engineering** - Sommerville, 7th edition, Pearson Education. [UNITS-2]

3. The unified modeling language user guide - [unit-3 half]
Grady Booch, James Rambaugh, Ivar Jacobson, Pearson Education.

Software Engineering Definitions :

The seminal definition:

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

IEEE Definition :

Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software

IMPORTANCE:

* The software technology plays an important role from the past three decades making world as digital with its rapid emergent technology.

* The software is the youngest industry compared with other industries, but now it is nucleus for other industries.

- The economies of all developed nations are dependent on software.
- More and more systems are software controlled (transportation, medical, telecommunications, military, industrial, entertainment)

- Software engineering is concerned with theories, methods and tools for professional software development.
- Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.
- Software costs **more to maintain** than it does to develop. For systems with a long life, maintenance costs may be several times development costs.
- Software engineering is concerned with cost-effective software development.
- The Objective of the Software Engineering is committed to build the software projects within the budget, time and required quality.
- The software engineering discipline that focus on the cost effective development of high quality software.
- Many software organizations have been seriously concentrating on effective ways to improve the product quality.
- The quality culture ultimately leads to development of more effective approaches in the software engineering.
- As many companies in the software industry still not following software engineering approaches effectively in their software development projects still produces software wi.th time delay, over budget and not realistic
- The main objective of Software Engineering is to produce the satisfactory systems on time and within the budget.
- The software product is intangible which consists of programs and associated documentation.
- Vast number of advanced process models, tools, methods has been developed to enable the software engineers in developing user friendly software products.
- The subject embedded concept Project Planning, Project Tracking, Formal Technical inspections, Configuration Management, Software Quality Assurance and Risk Management

THE EVOLVING ROLE OF SOFTWARE

- Software takes on a dual role. It is both product & a vehicle for delivering other products.

- As the vehicle for delivering the other products. Ex : Operating Systems, Network s/w, Software tools, and environments.

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

Hardware : Dramatic improvements in H/w performance and profound changes in computer architectures, vast increases in memory and storage capacity, Wide variety of exotic I/O options with sophisticated and complex computer based systems.

- Popular books published during (1970-80) provide useful historical insight into the changing perception of computers/software and impact on culture.

- **"New Industrial Revolution"** : . Toffler, Naisbitt, feigenbaum, McCorduck, Stoll.

- In 1990, Toffler described as " power shift" in which old organizations (govt, educational, industrial, economical and military) disintegrate as computers and software lead to "democratization of knowledge".

- Yourdon worried that US companies might lose their competitive edge related businesses and predicted"

- Today, a huge software industry has become a dominant factor in the economies of the industrialized world.

SOFTWARE PRODUCTS

- **Generic products**

Stand-alone systems that are marketed and sold to **any customer** who wishes to buy them.

Examples – PC software such as editing, graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.

- **Customized products**

Software that is commissioned by **a specific customer** to meet their own needs.

Examples – embedded control systems, air traffic control software, traffic monitoring systems.

SOFTWARE : Software is

- Set of instructions (i.e Computer Programs) – Provide desired feature, function, performance
- Data structures that enable the programs – adequately manipulate the data
- Documents that describe the operations & use of programs.

- The characters of software are different than other than that the human build.
- Software is logical than a physical system.

Characteristics :

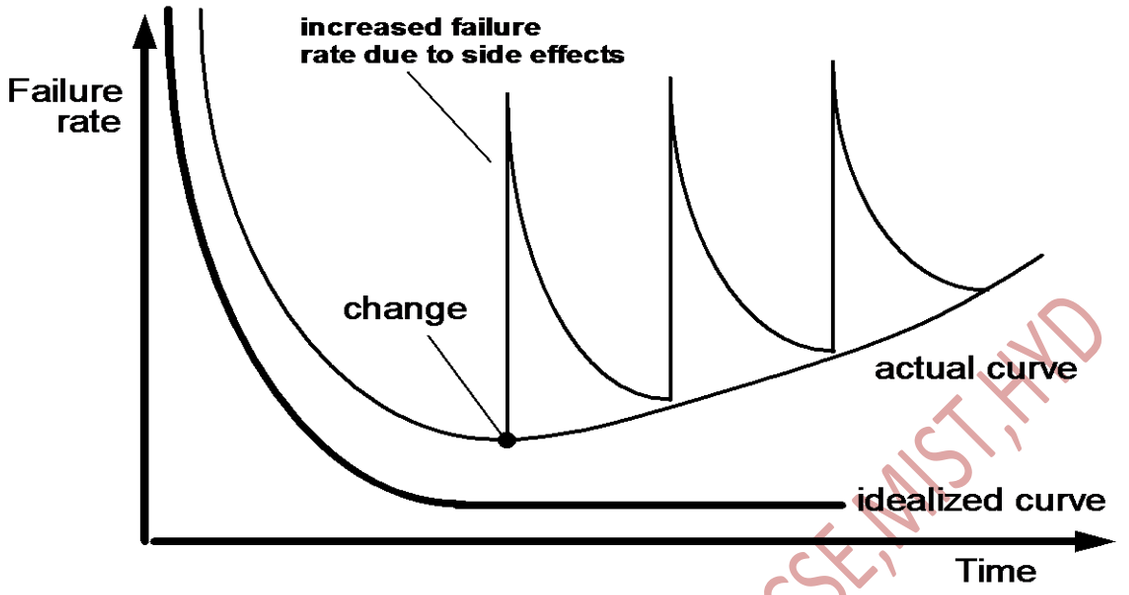
1. ***Software is developed or engineered , it is not manufactured in the classical sense.***

- **Software** [developed/engineered **]** and **Hardware [**manufactured)
- High quality achieved through the good design
- The both activities depends on people but relationship between people applied /& work accomplished is entirely different.
- Bothe activates require construction of a "product" , but approaches are different.

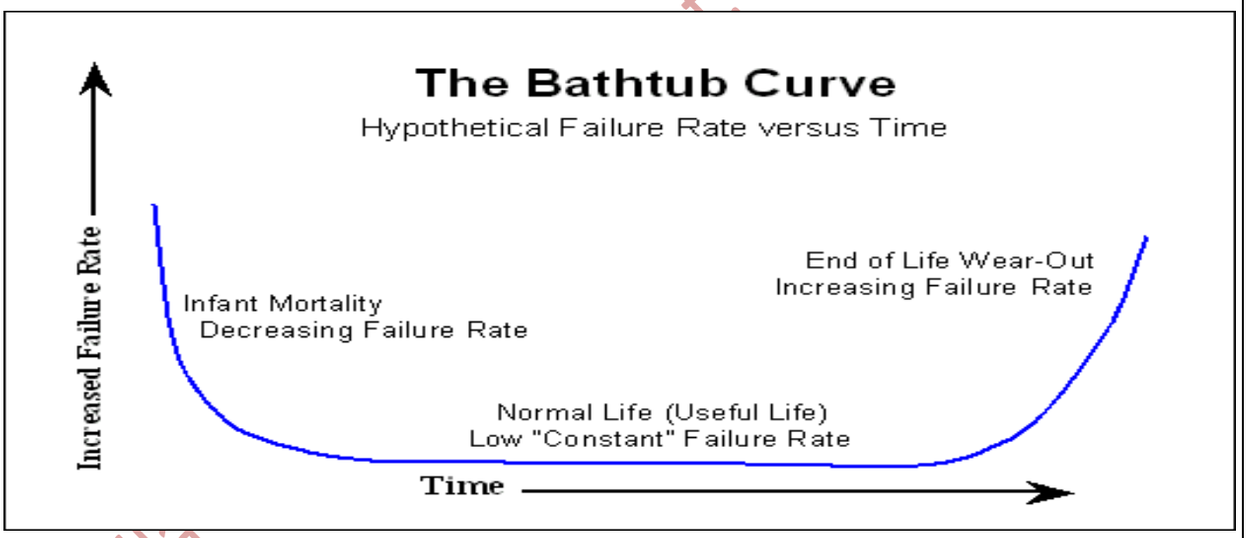
2. Software Doesn't Wear out.

SOFTWARE	HARDWARE
- Not change with environmental maladies. - High failure rate at early stage - Does n't Wear out, - Maintenance always needed - Achieved through Machine Executable code. - Maintenance of S/W is more complexity than H/W - Maintenance cost S/W>H/W - **Idealized Curve**	- .High failure rate in its life (design / manufacturing defects) - Defects are corrected, and failure rate drops to steady- state level) (hopefully go-slow) for period of time - As time passes the failure rate rises again as h/w components suffer from - Cumulative effect of dust, vibration, abuse ,temperature other environmental maladies. Simply **Hardware ware out.** - **Bathtub curve** - Hardware – replaced with spare parts

Idealized Curve [SOFTWARE] :



Bathtub Curve [HARDWARE]



3. ***Although the industry is moving toward component –based construction, most software continues to be custom built.***

- Modern reusable components encapsulate data and processing into software parts to be reused by different programs. E.g. graphical user interface, window, pull-down menus in library etc.

Hardware *designed, built, catalog, off shelf*

Software- *Reusable, custom-built, encapsulated , user interface, interfaces contains libraries*

2. THE CHANGING NATURE OF SOFTWARE :

System software : System software is collection of programs writer to service other programs.

Ex : Compilers, editors, and file management utilities,

Other – O/.S components, drivers, networking software, telecommunication processors.

- Heavy interaction with hardware.
- Heavy usage by multiple users, concurrent operation that require scheduling.
- Resource sharing, process management, multiple interfaces.

Application software :

- Stand alone programs that solve specific business need.
- Ex : Conventional data processing applications,
- application software is used to control the business functions in real-time.

Engineering/ Scientific software :

- Crunching algorithms for scientific applications
- Engineering and scientific algorithms range from astronomy to volcanology
- Conventional numerical algorithms
- Computer –aided design , system simulation and other interactive applications. Automotive Stress Analysis, Molecular Biology, orbital dynamics etc.

Embedded Software :

- Embedded software resides within a product or system
- Used to implement and control functions for the end user and system itself.
- Embedded software can perform limited and esoteric functions
- Ex : Keypad control of a microwave oven, digital functions of dash board display in car.

Product – line software :

- Designed to provide a specific capability for use by many different customers.
- Product –line software focus on limited and market place.
- Ex : Inventory control products, word-processing, graphics, multimedia, entertainment, database management financial, personal.

Web-applications :

- **"Webapps"** span a wide array of applications.
- Linked with hypertext files that present in formation using test and limited graphics.
- Webapps – sophisticated computing environments not only provide standalone features, computing functions,
- Integrated with databases and business applications.
- Ex: E-commerce and B2B applications

7. **Artificial Intelligence [AI] :**

- AI Software uses non-numerical algorithms to solve complex problem
- Robotics, expert system, pattern recognition [Image/voice], game playing

8. **Ubiquitous Computing :**

- The rapid growth of wireless networking may soon lead to true distributed computing.
- The ubiquitous computing allows small devices, personal computers and enterprise system to communicated across vast networks.

9. **Net sourcing :**

- The WWW is rapidly becoming a computing engine as well as content provider.
- Ex : Search Engines.

10. **Open Source** : - Self descriptive, free available developed by third party.

LEGACY SOFTWARE :

Older programs – often referred by "**Legacy Software**".

Outdated but Still usable.

The quality of Legacy Software:

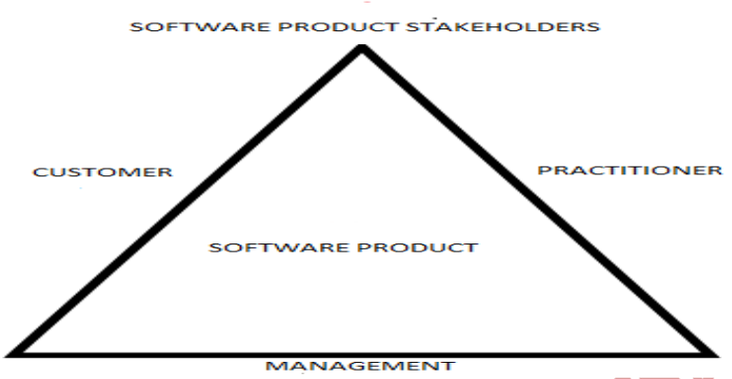
- Poor quality.
- ie. Inextensible designs, convoluted code, poor and nonexistent documentation, Test cases and results were never achieved.

Legacy S/W Usability:

- Adapted to meet the needs of new computing environments.
- Enhanced to implement new business requirements.
- Extended to make it interoperable with more modern systems/databases.
- Re-architected to make it viable with a network environment. .

SOFTWARE MYTHS :

- Erroneous Belief about S/W & Process used to built it , Myths have a no.of attributes.
- False expectations.
- Myths to be reasonable statements,
- Myths are misleading attributes (beliefs) that caused serious problems for managers, Technical people and other practitioners



- Classified into 3 types :

  1. Management Myths 2. Customer Myths 3. Programmer / Practitioner Myths

1. MANAGEMENT MYTHS:

- S/W Management is Responsibility and maintains the disciplines.
- Management is always under pressure with to maintain budgets, time schedules to improve the quality.

Myth 1: ***We already have a book that full of stands & procedures for build software. We not provide my people with every thing need to known***

Reality : The book of standards may very well exist, but is it used ?

Is it adoptable ?, Whether practitioners knows ?.

In many cases answer is "No".

Myth 2 : ***If we get behind people. We can add more programmers and catch-up (Sometimes called i.e Mongolian horde concept).***

Reality : - Software is not Manufacture process.

- Adding more people in between the process leads to time delay.
- The people can be added in planned and Well co-ordinated manner

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

Reality : If an organization (Third party) does not understand internal operations (mange & control) of the organization.

- It leads to ambiguity in final project(i.e. without understanding internal operations).

CUSTOMER MYTHS:

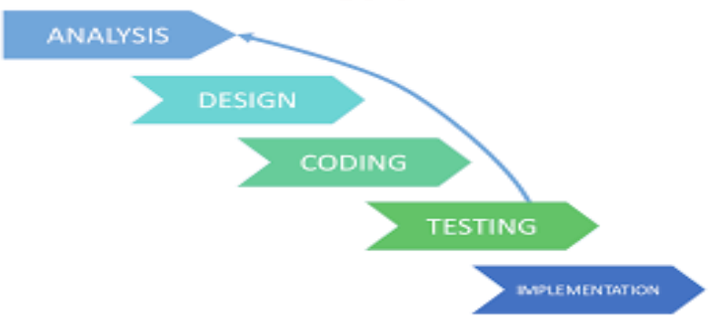
- Customer (end user) of the software who is other side desk.
- Customer believes myths about software.
- Myths leads to false expectations and ultimately leads dissatisfaction with developer.

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

Reality : - Comprehensive and stable of requirements is not always possible
- Ambiguous and partial requirements leads to ambiguous programming
- Unambiguous requirements leads to effective programming.
- It is possible through Effective & Continuation communication between customer & developer.

Myth 2 : *Project requirement continually change, but change can be easily accommodated because software is flexible.*

Reality : - True, Software requirements continually change.
- But Cost of insertion vary with time at the different phases of the development.
- Requirements -> Design - > Coding -> Test - > Implementation



PRACTITIONER'S MYTHS : Myths that still believed by software practitioners.

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

Reality : - The sooner you begin writing code, the longer it will take you to get done.
- 60% to 80% of all efforts are spent after software is delivery.

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

Reality : - One of the most software mechanisms such as Formal Technical Reviews are Quality filters to find the errors.
- Quality filters are more effective than testing for finding certain classes of software errors.

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

Reality : - A working program is only one part of software configuration that includes many elements.

- It needs Guidance and Software support.

Myth 4:***Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.***

Reality : - Software engineering is not only creating documents.

- It is for creating for better quality.
- The better quality reduced work and motivate fast delivery**.**

* Many software professional recognize the fallacy of software myths.
* Regrettably, habitual and methods, poor management and technical practices.

GENERIC VIEW OF PROCESS :

- Iterative learning process.
- Knowledge is collected, distilled, and organized as process
- Software process is frame work
- The process contains methods & Automated tools
- Software Engineering is layered technology.

SOFTWARE LAYERED TECHNOLOGY :

- Any Engineering approach must rest on organizational commitment to **quality** which fosters a continuous process improvement culture.
- Software Engineering is Systematic, disciplined, Quantifiable & adaptable approach.
- The software product should develop within the time, budget and with more quality, reliable for customer satisfaction.

Software Layered Technology consists with four layers

1. Quality Focus 2. Process 3. Methods 4. Tools

QUALITY FOCUS :

- Quality focus is Bedrock of Software Layered Technology
- Quality measured with Total Quality Management (TQM), Six Sigma methods and Statistical Methods

PROCESS LAYER:

- Software Engineering is set of framework with activities for effective delivery of software.
- Establish the context where products (model, data, report, and forms) are produced,
- Milestone are established, quality is ensured and change is managed.
- The process forma the basis for management control of software projects.

METHODS LAYER :

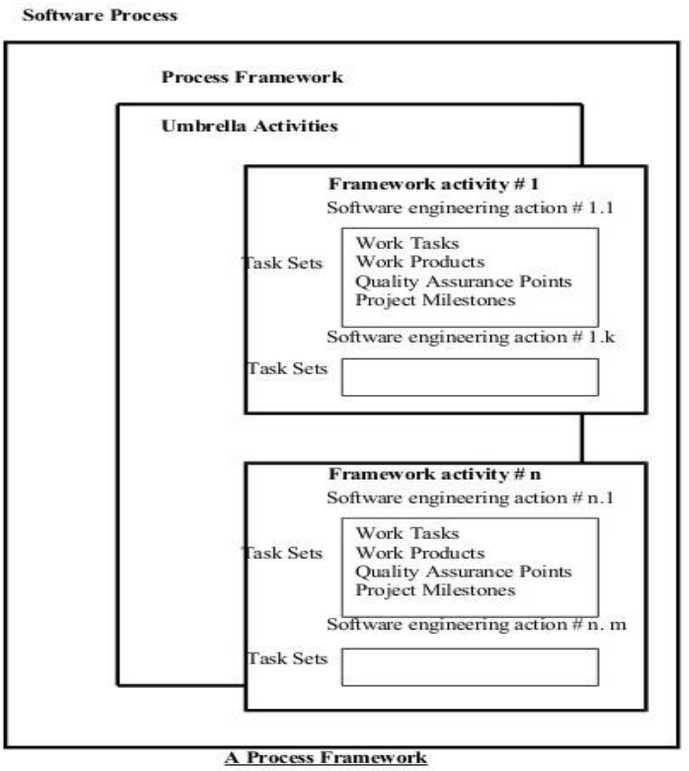
- Methods provides technical " how to's" for building software process.
- It encompasses many tasks such as communication, requirement analysis, design modeling, program construction, testing and support.
- Methods rely on set of basis principles, activities and techniques

TOOLS LAYER :

- Provide automated or semi-automated support for the process and methods.
- Management uses the process tools. and Methods uses different tools at the various phases of the software development.
- Example : CASE (COMPUTER AIDED SOFTWARE ENGINEERING)

SOFTWARE PROCESS FRAMEWORK :

- A Process framework is a foundation for complete software process.
- Identify a small No.of frame work activities that is applicable for all project regardless its size, complexity.
- A process is a collection of activities, actions and tasks that are performed when some work product is to be created. It is **not a rigid prescription** for how to build computer software. Rather, it is an adaptable approach that enables the people choose **appropriate set of work actions** and tasks.
- Purpose of process is to deliver software product in a timely manner and with required quality. It will satisfy the customer who will use it
- A Frame work is a umbrella activities.



A Process Framework

Framework activities :

Communication : - Communication between customer & Developer

- Set of Frame work activities.
- Requirement gathering , other related activities

Planning : Plan for

Overcome the Risks, mobilize the Resources,

Fix the Work Schedules and Work Products.

Modeling : - Models for better understanding .

- Full fil the software requirements.

Construction : - Code Generation, Testing, (Manual & Automated)

Deployment : - Evaluation , Feed back

Task sets : - Work products, millstones, schedules, quality assurance

- These five framework activities involved in software development regardless application domain, size of the project, complexity of the efforts etc, though the details will be different in each case.

- For many software projects, these framework activities are applied **iteratively** as a project progresses. Each iteration produces a software increment that provides a subset of overall software features and functionality.

UMBRELLA ACTIVITIES :

The five process framework activities and help team manage and control progress, quality, change, and risk.

Software project tracking and control : - Assess progress against the plan and take actions to maintain the schedule.

Risk management : Assesses risks that may affect the outcome and quality.

Software quality assurance : Defines and conduct activities to ensure quality.

Technical reviews : Assesses work products to uncover and remove errors before going to the next activity.

Measurement : Define and collects process, project, and product measures to ensure stakeholder's needs are met.

Software configuration management: Manage the effects of change throughout the software process.

Reusability management : Defines criteria for work product reuse and establishes mechanism to achieve reusable components.

Work product preparation and production: Create work products such as models, documents, logs, forms and lists.

Modeling :

Analysis : Requirement Gathering , Elaboration , Negotiation, specification, validation validation

Design : Data Design, Architectural Design, Interface Design, Component. Design,

Prescriptive process models stress detailed definition, identification, and application of process activates and tasks. Intent is to improve system quality, make projects more manageable, make delivery dates and costs more predictable, and guide teams of software engineers as they perform the work required to build a system.

- Unfortunately, there have been times when these objectives were not achieved

Agile process models emphasize project "agility" and follow a set of principles that lead to a more informal approach to software process. It emphasizes maneuverability and adaptability. It is particularly useful when Web applications are engineered.

- Quick analysis , Design and implementation.

International Standard Organization [ISO] :

- Describes an evolutionary improvement path for software organizations from an ad hoc, immature process to a mature, disciplined one.

- Provides guidance on how to gain control of processes for developing and maintaining software & how to evolve toward a culture of software engineering and management excellence.

- It is the world's leading developer of International Standards.
- It has 156 member countries.
- Its portfolio holds more than 15,036 standards that are used in every sector of business, industry and technology.

- Created by the Software Engineering Institute, a research center founded by Congress in 1984.
- A structure designed to direct IT organizations through software process improvement
- Philosophy of "continuous process improvement".

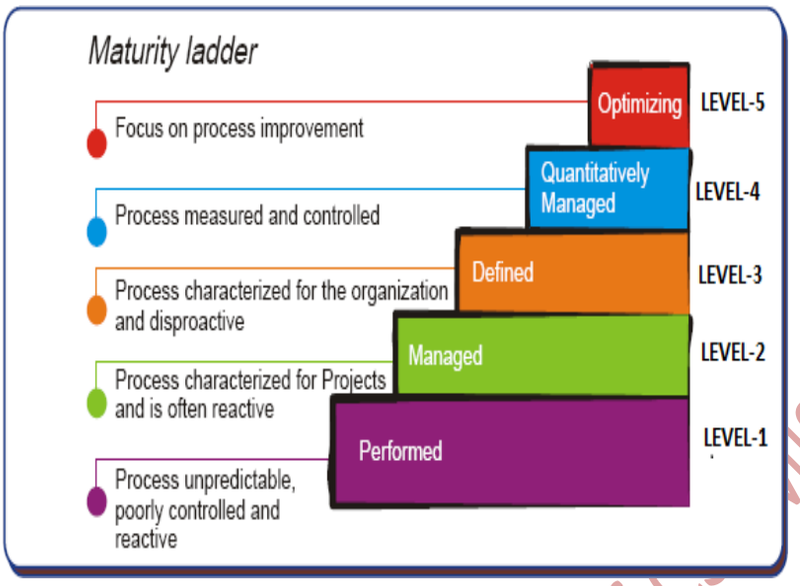
CAPABILITY MATURITY MODEL INTEGRATION [CMMI] :

- Software Engineering Institute (SEI) developed comprehensive meta model.
  - Software Process Management Maturity.
  - Gauge the Organization Capability & Maturity.
  - The Meta Model in two different ways
    1. Continuous Model 2. Stayed Model
  - Each process is formally assessed against specific goals and practices and is rated according to the following capability levels.
  - Software Process Performance
    - Actual results achieved by following a software process.
  - Software Process Maturity
    - Extent to which a specific process is explicitly defined, managed, measured, controlled and effective.
    - Implies potential growth in capability
    - Indicates richness of process and consistency with which it is applied in projects throughout the organization.

Maturity level indicates level of process capability :

- LEVEL 0 [INCOMPLETE]
- LEVEL 1 [PERFORMED]
- LEVEL 2 [MANAGED]
- LEVEL 3 [DEFINED]
- LEVEL 4 [QUANTITATIVELY MANAGED]
- LEVEL 5 [OPTIMIZED]

CAPABILITY MATURITY MODEL INTEGRATION [CMMI]



LEVEL 0 : [INCOMPLETE]

The process area [Ex :Requirement Management -RE] is either not performed or doesn't achieve all the goals and objectives defined by CMMI for level 1 Capability.

LEVEL 1 : [PERFORMED]

All the specific goals of the process area (as defined by CMMI) have been satisfied. Work tasks required to produce work products are being conducted.

- The software process is characterized as ad hoc, even chaotic.

LEVEL 2 : [MANAGED]

- Process area confirms to an organizationally defined policy
- All people doing the work have to adequate resources to get the job done.
- Stake holders involved in the process area.
- Monitored , Controlled , Reviewed and evaluated.
- Process management is well established to track cost, schedule, and functionality. .

LEVEL 3 : [DEFINED]

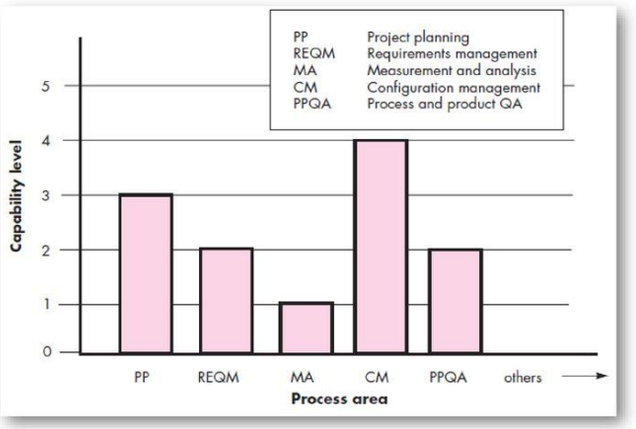
- Tailored form the organization's set of standard process according to the organization's Tailoring guidelines.
- Contribute work products, measures, and other process improvement.
- The management and engineering activities is documented, standardized, and integrated into a standard software process for the organization.

LEVEL 4: [QUANTITATIVELY MANAGED]

- Measures of the software process and product quality are collected.
- Software process and products are quantitatively understood and controlled.
  * Quantitative Assessment * Process performance

LEVEL 5 : [OPTIMIZED] : - Change management process / technology

- Adapted and optimized using quantitative (statistical) means to meet customer needs to continually improve efficiency.
- Continuous process improvement is enabled by quantitative feedback.
- Adopted innovative ideas and technologies .
- Cost benefit analysis of new technologies and proposed process changes.



CMMI process area capability profile

19

SPECIFIC / GENERIC GOALS & PRACTICES :

SG-1 : Establish estimates.

SP1.1. Estimate the scope of the project.

SP1.2. Establish estimates of work product and task attributes.

SP1.3. Define project life cycle.

SP1.4 Determine estimates of effort and cost.

SG-2. Develop a Project Plan

SP2.1. Establish the budget and schedule.

SP2.2. Identify project risks.

SP2.3. Plan for data management.

SP2.4. Plan for project resources.

SP2.5. Plan for needed knowledge and skills

SP2.6 Plan stakeholder holder involvement.

SP2.7 Establish the project plan.

SG-3 : Obtain commitment to the plan.

SP3.1 Review plans that affect the project.

SP3.2 Reconcile work and resource levels.

SP3.3. Obtain plan commitment.

GENERIC :

GG-1 : Achieve specific goals.

GP1.1. Perform base practices.

GG-2 : Institutionalize a managed process.

GP2.1. Establish an organizational policy.

GP2.2 Plan the process

GP2.3 Provide resources.

GP2.4 Assign responsibility

GP2.5 Train People

GP2.6 Manage Configurations.

GP2.7 Identify and involve relevant stakeholders.

GP2.8. Monitor and control the process.

GP2.9 Objectively evaluate adherence.

GP2.10 Review status with higher level management.

GG -3 : Institutionalize a defined process.

GP 3.1 Establish a defined process.

GP 3.2 Collect improvement information.

GG -4 : Institutionalize a quantitatively managed process.

GP 4.1 Establish quantitative objectives for the process.

GP 4.2 Stabilize sub process performance.

GG -5 : Institutionalize an optimizing process.

GP 5.1 Ensure continuous process improvement.

GP 5.2. Correct root causes of problems

PROCESS PATTERNS :

- A *process pattern*
  - describes a process-related problem that is encountered during software engineering work.
  -
  - identifies the environment in which the problem has been encountered, and
  - suggests one or more proven solutions to the problem.
  - Stated in more general terms, a process pattern provides you with a *template* [Amb98]—a consistent method for describing problem solutions within the context of the software process.

TYPES OF PATTERNS :

- *Stage patterns*—defines a problem associated with a framework activity for the process.

- *Task patterns*—defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice.

- *Phase patterns*—define the sequence of framework activities that occur with the process, even when the overall flow of activities is iterative in nature.

Process Assessment and Improvement :

* The existence of a software process is no guarantee that .
  - The software will be delivered on time
  - It will meet the customer's needs
  - It will exhibit the technical characteristics that will lead to long-term quality.

* Process can be assessed to ensure that it meets a set of basic process criteria
  - Process patterns must be coupled with solid software engineering.

Process Assessment and Improvement :

- **Standard CMMI Assessment Method for Process Improvement (SCAMPI) :** Provides a five step process assessment model that incorporates five phases: initiating, diagnosing, establishing, acting and learning.

- **CMM-Based Appraisal for Internal Process Improvement (CBA IPI) :** Provides a diagnostic technique for assessing the relative maturity of a software organization; uses the SEI CMM as the basis for the assessment

- **SPICE—The SPICE (ISO/IEC15504) :** standard defines a set of requirements for software process assessment. The intent of the standard is to assist organizations in developing an objective evaluation of the efficacy of any defined software process.

- **ISO 9001:2000 for Software :** A generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides. Therefore, the standard is directly applicable to software organizations and companies. [Ant06]

Personal Software Process (PSP) :

- Planning. This activity isolates requirements and develops both size and resource estimates. In addition, a defect estimate (the number of defects projected for the work) is made. All metrics are recorded on worksheets or templates. Finally, development tasks are identified and a project schedule is created.

- High-level design. External specifications for each component to be constructed are developed and a component design is created. Prototypes are built when uncertainty exists. All issues are recorded and tracked.

- High-level design review. Formal verification methods (Chapter 21) are applied to uncover errors in the design. Metrics are maintained for all important tasks and work results.

- Development. The component level design is refined and reviewed. Code is generated, reviewed, compiled, and tested. Metrics are maintained for all important tasks and work results.

- Postmortem. Using the measures and metrics collected (this is a substantial amount of data that should be analyzed statistically), the effectiveness of the process is determined. Measures and metrics should provide guidance for modifying the process to improve its effectiveness

Team Software Process (TSP) :

- Build self-directed teams that plan and track their work, establish goals, and own their processes and plans. These can be pure software teams or integrated product teams (IPT) of three to about 20 engineers.

- Show managers how to coach and motivate their teams and how to help them sustain peak performance.

- Accelerate software process improvement by making CMM Level 5 behavior normal and expected.
  - The Capability Maturity Model (CMM), a measure of the effectiveness of a software process, is discussed in Chapter 30.
  - Provide improvement guidance to high-maturity organizations.

- Facilitate university teaching of industrial-grade team skills

PROCESS MODELS

Prescriptive process models :

- Prescriptive process models advocate an orderly approach to software engineering.
- Focuses on structure, order & project consistency in software development .
- Define a prescribed set of process elements and a predictable process work flow.

- Every S/W organization should describe a unique set of frame work activities for software process. Each frame work activity with set of S/E actions. Each action in terms of a task set that identifies the work to be accomplished to meet the goals.

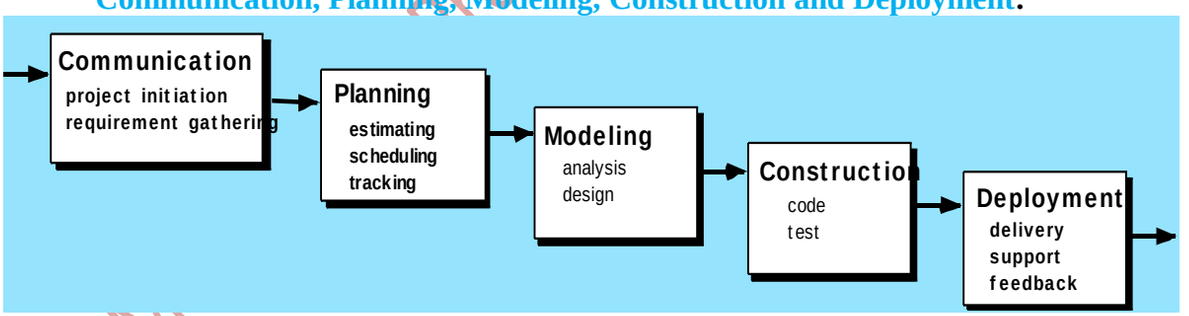
  Framework activity – S/w Engg actions – task set – goals.

- Frame work activities –Communication, Planning, Modeling, Construction, , Deployment.
- Apart from the framework activities the other activities like Q.A, Config.Mgnt, Control mechanism
- The manner in which the process elements are interrelated to one another

WATERFALL MODEL :

- A classic life cycle which suggests a systematic, sequential approach to software development. It is Linear fashion, Sequential life cycle, Classic Life Cycle.
- Systematic Approach, Oldest Paradigm, Traditional Life cycle.
- Starts with customer specifications of requirements and progresses with planning, modeling , construction and concluded with deployment.

Communication, Planning, Modeling, Construction and Deployment.



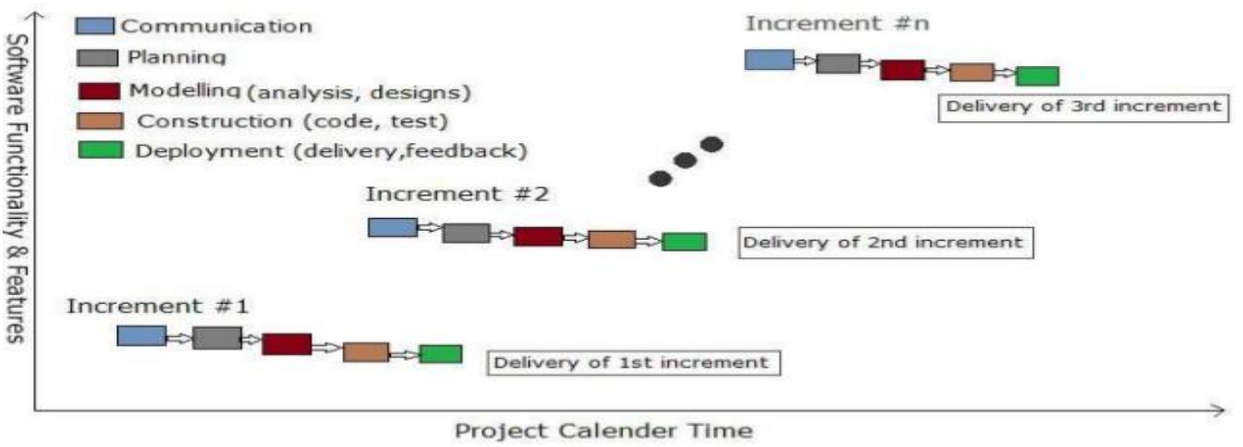
Drawbacks:

- Real time Projects rarely follow. Although the linear model can accommodate iteration. - Changes can cause confusion
- It is difficult for the customer to state all requirements explicitly. Requirements are not reveled initially, it is problematic to accommodate in mean time.
- The stake holder should patience. A working version of the program will not be available until late in the project time span.
- Blacking states. – (Some of the team members should wait for tasks of other members should complete).
- However, it can serve as a useful process model in situations where requirements are fixed and work is to proceed to completion in a linear manner.

INCREMENTAL MODEL :

- The incremental model elements Combine the Waterfall Model - Iteration fashion.
- The incremental model applies linear sequence in staggered fashion in Calendar time progress.
- Each Linear sequence produces -" deliverable increments".
- First increment is core product. The remaining are supplementary features.
- The Progress continues until the complete product produced.
- Incremental model like prototyping and other evolutionary approaches are iterative nature.
- Operational product in each increment.

Tasks in Incremental Model

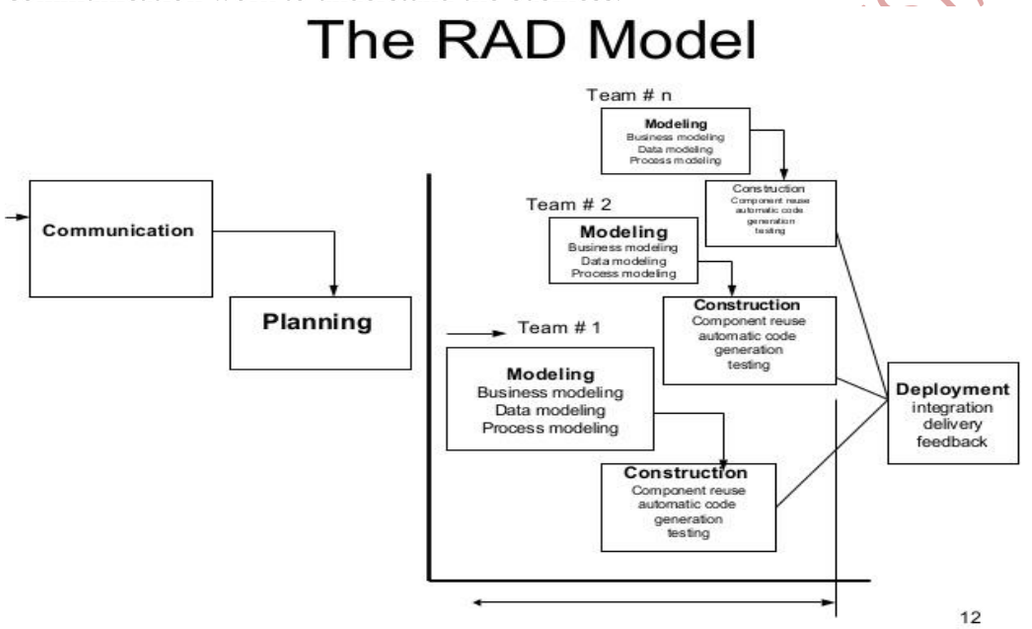


Advantages :

- Incremental process model is particularly useful when staffing is unavailable for complete implementation by the business deadline that has been establish for the project.
- Early increments can be implemented with fewer people. If the core product is well received, additional staff can be added to implement the next increment.
- Increments can be planned to manage the technical risks.

RAPID APPLICATION DEVELOPMENT MODEL [RAD] :

- The Rapid Application Development (RAD) is an incremental software process model that emphasizes a short development cycle.
- RAD Model is “high speed” adaption of water fall model., in which rapid development is achieved by using a component based construction approach.
- If the requirements are well understood project scope is constrained.
- RAD process enables a development team to create a “ fully functional system” within very short time period .
- The RAD model will also contains the frame work activities like other models.
- The communication work to understand the business.



- The **planning is** made for the multiple software teams work in parallel on different s/w teams.
- Modeling encompasses three major phases.
  *Business Modeling * Data Modeling * Process modeling.
- Construction emphasizes the use of pre existing software components and the application of automatic code generation. Finally deployment establishes a basis for subsequent iterations.
- **Development time** 60-90 days.
- Each major function can be addressed by a separate RAD teams and then **integrated as whole.**

Advantages & Disadvantages :

- **For Large Scalable projects** , the RAD requires more man power needed to form right No.of RAD teams.
- Developers & Customers are not committed, it is not possible to complete in within the time frame.
- If the **system cannot be properly modularized**, building the components necessary for RAD will be problematic.
- **Performance** is to be achieved through tuning the interfaces to system components, the RAD approach may not work.
- RAD may not be appropriate when technical risk are high (e.g when a new application makes heavy use of new technology).

EVOLUTIONARY PROCESS MODELS :

- Software evolves over a period of time.
- Business & product requirements often change as development proceeds.
- Making a straight line path to an end product unrealistic.
- A set of core product or system requirements is well understood, but the details of product or system extensions have yet to be defined.
- We need a process model that has been explicitly designed to accommodate a product that grows and changes .
- Produce an increasingly more complete version of the software with each iteration .
- What is the difference b/w incremental models and evolutionary models?

  Incremental models : each pass produces simply increments .
  Evolutionary models : each pass produces more complete version of S/W

PROTOTYPE MODEL :

Communication : You meet with other stakeholders to define the overall objectives, identify known requirements, and outline areas where further definition are mandatory.

Quick plan : A prototyping iteration is planed quickly.
Modeling :
- Quick design
- Focuses on a representation of those aspects of the S/W that will be visible to end users.

Construction of prototype :
Deployment, delivery & feedback :
The prototype is deployed and evaluated by stakeholders, who provide feedback that is used to further refine requirements

- Prototype is made first and based on it final product is developed.
- A prototype is a model or a program which is not based on strict planning, but is an early approximation of the final product or software system.
- A prototype acts as a sample to test the process. From this sample we learn and try to build a better final product.
- Please note that this prototype may or may not be completely the final system we are trying to develop.

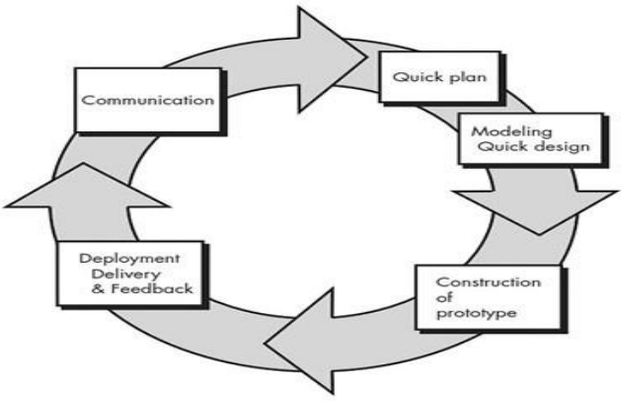


Figure: Prototype Model

Advantages :

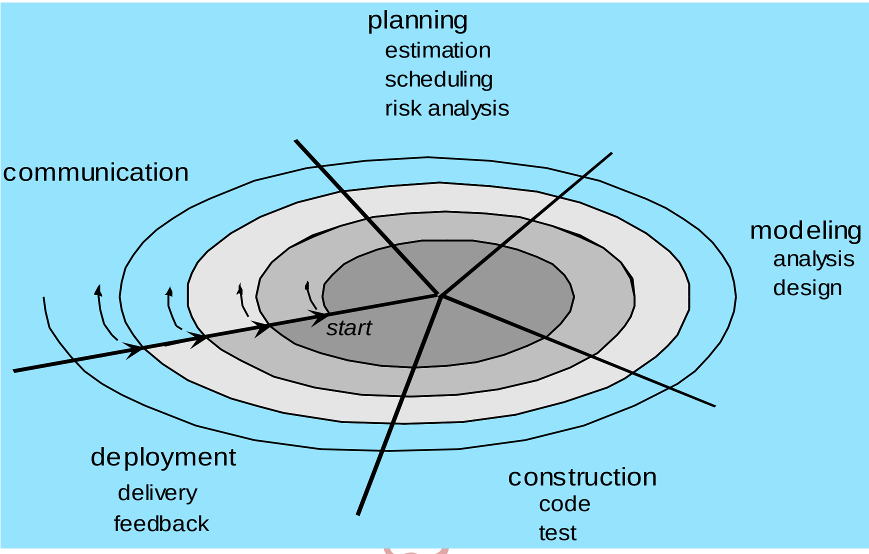
- When prototype is shown to the user, he gets a proper clarity and 'feel' of the functionality of the software and he can suggest changes and modifications.
- This type of approach of developing the software is used for non-IT-literate people.
- When client is not confident about the developer's capabilities, he asks for a small prototype to be built. Based on this model, he judges capabilities of developer.
- Sometimes it helps to demonstrate the concept to prospective investors to get funding for project.
- It reduces risk of failure, as potential risks can be identified early and mitigation steps can be taken.
- Iteration between development team and client provides a very good and conductive environment during project.

Disadvantages :

- Prototyping is usually done at the cost of the developer. So it should be done using minimal resources. It can be done using Rapid Application Development (RAD) tools.
- Once we get proper requirements from client after showing prototype model, it may be of no use. That is why, sometimes we refer to the prototype as "Throw-away" prototype.
- It is a slow process.
- Too much involvement of client, is not always preferred by the developer.
- Too many changes can disturb the rhythm of the development team.

THE SPIRAL MODEL :

- Spiral Model proposed by Boehm.
- Evolutionary Software model, Couples the iterative nature of prototyping with controlled and systematic aspects of water fall model.
- It provides the potential for rapid development, yields more complete versions of s/w.
- Software developed in series of evolutionary releases.
- The spiral model divided into set of framework activities.
- Each framework activity represents one segment of the spiral path.



- As Evolutionary process begins, S/W teams perform activities that are implied by circuit around the spiral clock wise direction begins at centre.
- Risk is considered as each revolution is made.
- Anchor Milestones - a combination of work products and conditions that are attained along the path of the spiral.
- The first circuit around spiral might result in development of a product specification, the *subsequent passes* around the spiral might be used to develop a prototype and then progressively more sophisticated version of software.

- The first circuit around the spiral might represent a "concept development project", which starts at core of spiral and continuous for multiple iterations until the concept development is completed.
-
- New product development project - > spiral model is project enhancement project.

Advantages :

- The spiral model is realistic approach to the development of large-scale software systems.
- Each revolution is risk reduction.
- The Spiral model is systematic stepwise approach suggested by the classic life cycle, but incorporates into an iterative framework. That more realistically reflects the real world.

SPECIALIZED PROCESS MODELS

Component –Based Development :

- Commercial off the shelf (COTS) software components developed by venders, who offer them as products.
- These components provide targeted functionality, with well defined interfaces that enable the components to be integrated into the software.
- The components based development model incorporates many of the characteristics of Spiral model.
- It evolutionary in nature, iterative approach to the creation of software. These are pre packed software components.
- Modeling and construction activities begin with the identification of candidate components. Ex : Object oriented classes or packages .

Incorporate the following steps :

- Available component based products – researched and evaluated
- Component integration issued considered
- A software architecture is designed to accommodate components.
- Comprehensive testing is conducted for functionality

Features :

- Reuse - Reusability leads to measureable benefits.
- 70% development time , 84% - cost,

Aspect Oriented Software development :

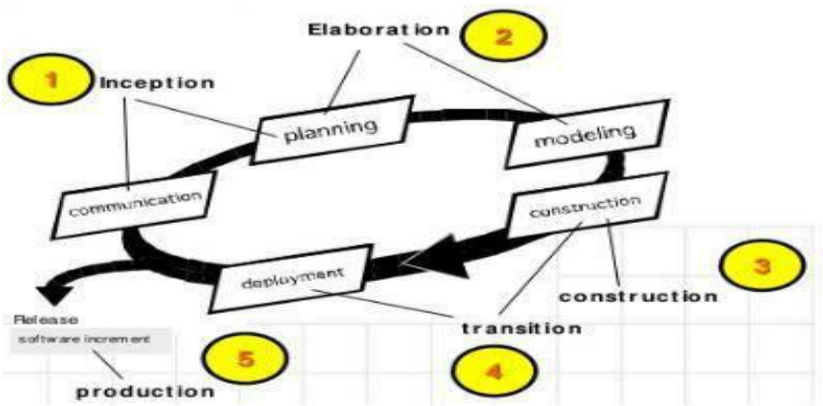
- The aspect-oriented programming [AOP] provides modularization of software systems by encapsulating cross cutting concerns.
- The cross cutting concerns affects several classes or modules.
- The concerns cut across multiple system functions, features.
- The aspects encapsulate behaviors that affect multiple classes into reusable modules. The aspects for protecting the modularity, transactions, security, authorization, exception handling, transaction management, optimistic concurrency, etc
- In the aspect oriented programming the code scattering and tangling can be avoided. The cross-cutting functionality is now encapsulated within well defined modules.

UNIFIED PROCESS :

- 1980-1990 Object oriented methods and programming language.
- A wide variety of object-oriented analysis and design (OOAD) methods were proposed during the same time period
- 199O James Raumbaugh , Grady Booch began work on a unified methods that would combine the OO features.
- The result was UML – A Unified modeling language that contain a robust notation of the modeling and development of OO systems.
- UML became an industry standard for object oriented software development.
- The Rational Corporation and other venders developed automated tools to support UML Methods.
- UML Provides the technology to support object-oriented software engineering practice, but it provides the framework to guide project teams in their applications of technology.
- The Unified modeling language is standard language for writing software blue prints.
- The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system.
- It is a very expressive language, addressing all the views needed to develop and then deploy such systems.
- The UML is one part of software development.
- The UML provides a vocabulary and the rules for combining words in that vocabulary for the purpose of communication.
- A Modeling language consisting the vocabulary and rules focus on the conceptual and physical representation of a system.

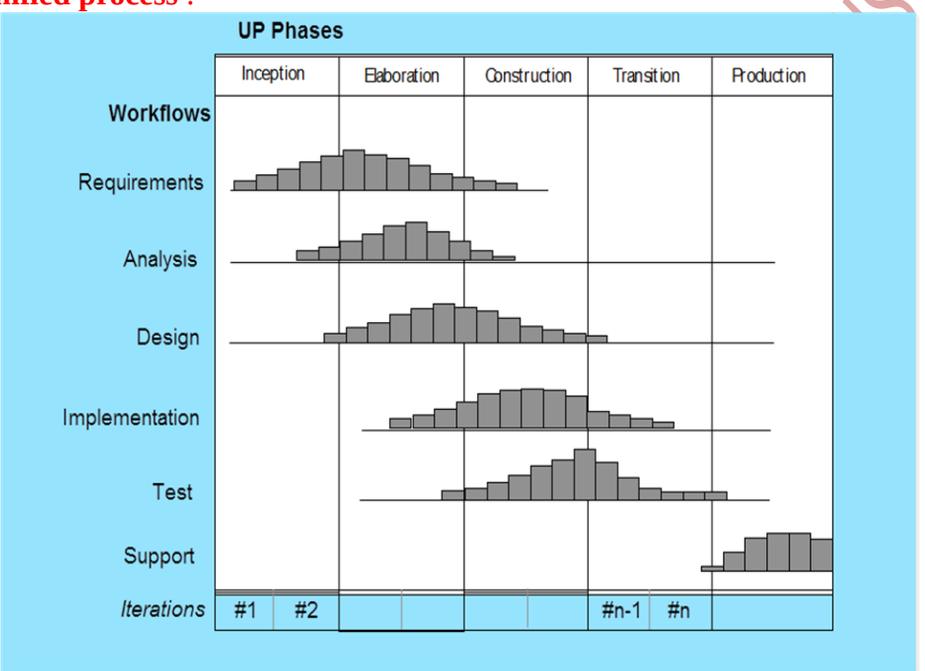
Characteristics :

- **Visualizing** : Algorithm is set of instructions [textually], which can't visualizing the problem graphically. The UML is graphical language which contain the bunch of graphical symbols. Each symbol in the UML notation is a well-defined semantics. We can express solution of the problem more unambiguously.
- **Specifying** : Specifying means that the models are more precise, unambiguous and complete. The UML addresses the specification of all the important analysis, design, and implementation stages of software intensive system.
- **Constructing** : The UML is not a visual language, but its models can be directly connected/ maps to such as java, C++, Visual Basic etc. The mapping permits forward engineering. The generation of code from a UML model into a programming language.
  .
- **Document** : The UML languages can supports the documentation of the software artifacts.



Unified process

Phases of Unified process :



Inception Phase: The inception is domain understanding.

Elaboration Phase:

- The elaboration phase encompasses the customer communication and modeling activities of the generic process model.
- The elaboration refines and expands the preliminary uses-cases , architectural representation in five different view of the software. i.e use-case model, analysis model, design model, implementation model and *deployment model.*
- The elaboration creates an " executable architectural baseline". The culmination of elaboration phase to ensure that scope, risks and delivery dates reasonable. The modification to the phase may be at this time.

Construction Phase : The construction phase of UP is defined to construction activity defined for the generic software process. he architectural model as input, the construction phase develops or acquires the software components that will make each use-case operation of end users. The required features and function of software increment are then implemented as source code. The Testing can be made (Assembly/ integration testing) made in this phase.

Transition Phase: The transition phase of the UP encompasses the later stages of the generic construction activity and first part of the generic deployment activity. The Software is given to end users for beta testing, and user feedback reports both defects and necessary changes. The software teams creates the support documentation for the software.

Production Phase : The production phase of UP with the deployment activity of the generic process..construction phase, transition, production phase that represent a "first cut" of the executable system .

IMPORTANT QUESTIONS

1. Explain the role and importance of Software Engineering.
2. Define software and its characteristics. *
3. Explain the nature of software. *
4. Explain the various myths of software engineering. *
5. Explain the about Legacy Systems, why legacy systems used.
6. Explain about the software Layered Technology.
7. Define the Software framework and its functionalities. *
8. Explain about Capability Maturity model and its significance. *
9. Explain the process patterns.
10. Explain about personnel software process & Team software process
11. Distinguish between the waterfall model and Incremental model. *
11. Explain about the Rapid Application Development model. *
13. Explain about the Evolutionary process models.
14. Explain about the Spiral Model with Advantages & Disadvantages. *
15. Explain about Unified Process Model and its phases.

Note : ('*' Mark questions are Assignment Questions)