UNIT - I

INTRODUCTION:

Software Engineering is a framework for building software and is an engineering approach to software development. Software programs can be developed without S/E principles and methodologies but they are indispensable if we want to achieve good quality software in a cost effective manner.
Software is defined as:

Instructions + Data Structures + Documents

Engineering is the branch of science and technology concerned with the design, building, and use of engines, machines, and structures. It is the application of science, tools and methods to find cost effective solution to simple and complex problems.

SOFTWARE ENGINEERING is defined as a systematic, disciplined and quantifiable approach for the development, operation and maintenance of software.

The Evolving role of software

The dual role of Software is as follows:

1. A Product- Information transformer producing, managing and displaying information.
2. A Vehicle for delivering a product- Control of computer(operating system),the communication of information(networks) and the creation of other programs.

Characteristics of software

- **Software is developed or engineered**, but it is not manufactured in the classical sense.
- **Software does not wear out,** but it deteriorates due to change.
- **Software is custom built** rather than assembling existing components.

THE CHANGING NATURE OFSOFTWARE

The various categories of software are

1. System software
2. Application software
3. Engineering and scientific software
4. Embedded software
5. Product-line software
6. Web-applications
7. Artificial intelligence software

- **System software.** System software is a collection of programs written to service other programs
- **Embedded software**-- resides in read-only memory and is used to control products and systems for the consumer and industrial markets.
- **Artificial intelligence software.** Artificial intelligence (AI) software makes use of nonnumeric algorithms to solve complex problems that are not amenable to computation or straightforward analysis
- **Engineering and scientific software.** Engineering and scientific software have been characterized

by "number crunching" algorithms.

LEGACY SOFTWARE

Legacy software are older programs that are developed decades ago. The quality of legacy software is poor because it has inextensible design, convoluted code, poor and nonexistent documentation, test cases and results that are not achieved.

As time passes legacy systems evolve due to following reasons:

- The software must be adapted to meet the needs of new computing environment or technology.
- The software must be enhanced to implement new business requirements.
- The software must be extended to make it interoperable with more modern systems or database
- The software must be rearchitected to make it viable within a network environment.

SOFTWARE MYTHS

Myths are widely held but false beliefs and views which propagate misinformation and confusion.
Three types of myth are associated with software:

- Management myth
- Customer myth
- Practitioner's myth

MANAGEMENT MYTHS

- Myth(1)-The available standards and procedures for software are enough.
- Myth(2)-Each organization feel that they have state-of-art software development tools since they have latest computer.
- Myth(3)-Adding more programmers when the work is behind schedule can catch up.
- Myth(4)-Outsourcing the software project to third party, we can relax and let that party build it.

CUSTOMER MYTHS

- Myth(1)- General statement of objective is enough to begin writing programs, the details can be filled in later.
- Myth(2)-Software is easy to change because software is flexible

PRACTITIONER'S MYTH

- Myth(1)-Once the program is written, the job has been done.
- Myth(2)-Until the program is running, there is no way of assessing the quality.
- Myth(3)-The only deliverable work product is the working program
- Myth(4)-Software Engineering creates voluminous and unnecessary documentation and invariably slows down software development.

SOFTWARE ENGINEERING-A LAYERED TECHNOLOGY

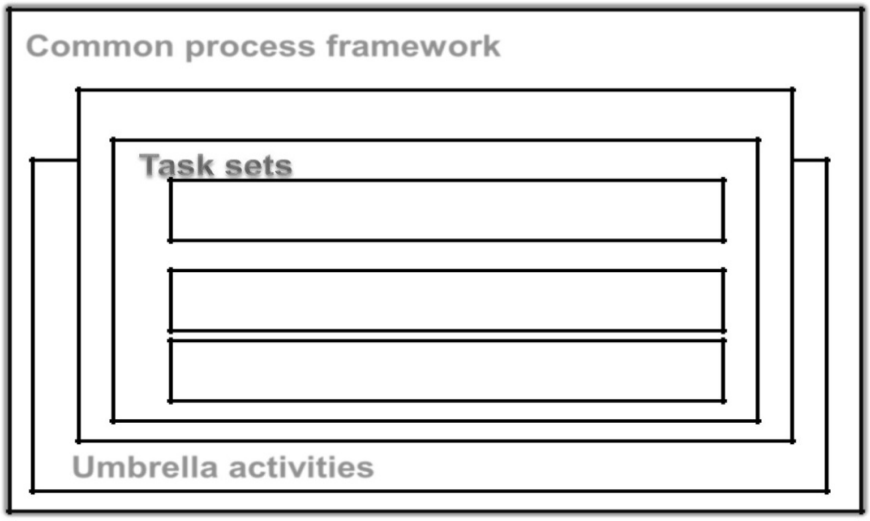
Fig: Software Engineering-A layered technology

SOFTWARE ENGINEERING - A LAYERED TECHNOLOGY

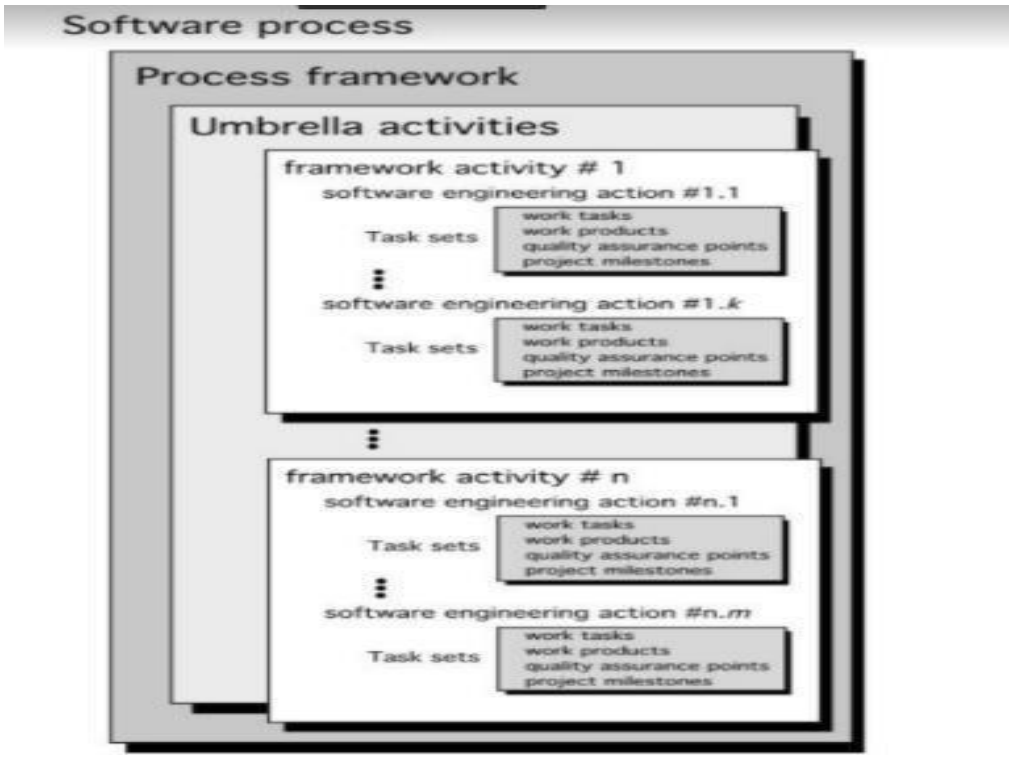
- Quality focus - Bedrock that supports Software Engineering.
- Process - Foundation for software Engineering
- Methods - Provide technical How-to's for building software
- Tools - Provide semi-automatic and automatic support to methods

A PROCESS FRAMEWORK

- Establishes the foundation for a complete software process
- Identifies a number of framework activities applicable to all software projects
- Also include a set of umbrella activities that are applicable across the entire software process.



A PROCESS FRAMEWORK comprises of :
Common process framework Umbrella activities Framework activities
Tasks, Milestones, deliverables SQA points



A PROCESS FRAMEWORK

Used as a basis for the description of process models Generic process activities

- Communication
- Planning
- Modeling
- Construction
- Deployment

A PROCESS FRAMEWORK

Generic view of engineering complimented by a number of umbrella activities

- Software project tracking and control
- Formal technical reviews
- Software quality assurance
- Software configuration management
- Document preparation and production
- Reusability management
- Measurement
- Risk management

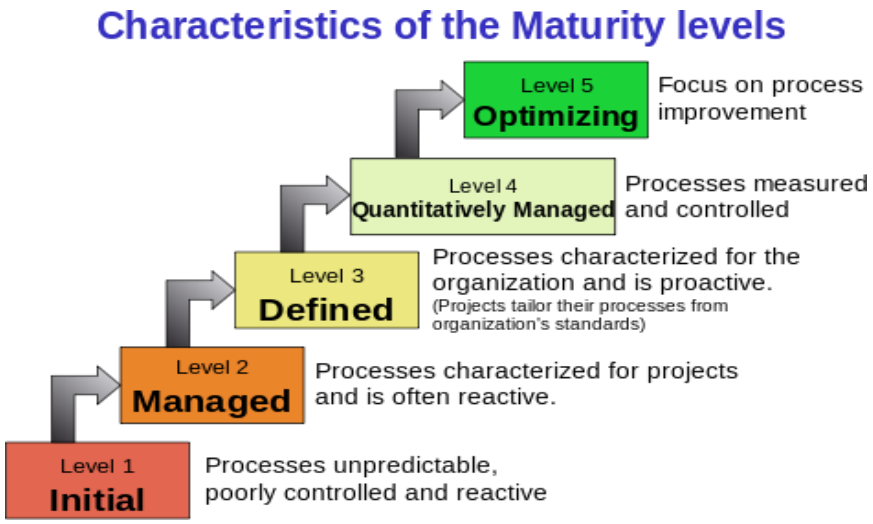
CAPABILITY MATURITY MODEL INTEGRATION(CMMI)

- Developed by SEI(Software Engineering institute)
- Assess the process model followed by an organization and rate the organization with different levels
- A set of software engineering capabilities should be present as organizations reach different levels of process capability and maturity.

CMMI process meta model can be represented in different ways
1.A continuous model
2.A staged model

Continuous model:
-Lets organization select specific improvement that best meet its business objectives and minimize risk-Levels are called capability levels.
-Describes a process in 2 dimensions
-Each process area is assessed against specific goals and practices and is rated according to the following capability levels.

CMMI
- Six levels of CMMI
– Level 0:Incomplete
– Level 1:Performed
– Level 2:Managed
– Level 3:Defined
– Level 4:Quantitatively managed
– Level 5:Optimized



CMMI
- Incomplete -Process is adhoc . Objective and goal of process areas are not known
- Performed -Goal, objective, work tasks, work products and other activities of software process are carried out
- Managed -Activities are monitored, reviewed, evaluated and controlled
- Defined -Activities are standardized, integrated and documented
- Quantitatively Managed -Metrics and indicators are available to measure the process and quality
- Optimized - Continuous process improvement based on quantitative feed back from the user

-Use of innovative ideas and techniques, statistical quality control and other methods for process improvement.

CMMI - Staged model

- This model is used if you have no clue of how to improve the process for quality software.
- It gives a suggestion of what things other organizations have found helpful to work first
- Levels are called maturity levels

PROCESS PATTERNS

Software Process is defined as collection of Patterns.Process pattern provides a template. It comprises of

- Process Template

-Pattern Name
-Intent
-Types
-Task pattern
- Stage pattern
-Phase Pattern

- Initial Context
- Problem
- Solution
- Resulting Context
- Related Patterns

PROCESS ASSESSMENT

Does not specify the quality of the software or whether the software will be delivered on time or will it stand up to the user requirements. It attempts to keep a check on the current state of the software process with the intention of improving it.

PROCESS ASSESSMENT

Software Process

Software Process Assessment Software Process improvement Motivates Capability determination

APPROACHES TO SOFTWARE ASSESSMENT

- Standard CMMI assessment (SCAMPI)
- CMM based appraisal for internal process improvement
- SPICE(ISO/IEC 15504)
- ISO 9001:2000 for software

Personal and Team Software Process

Personal software process

- PLANNING
- HIGH LEVEL DESIGN
- HIGH LEVEL DESIGN REVIEW
- DEVELOPMENT
- POSTMORTEM

Personal and Team Software Process

Team software process Goal of TSP

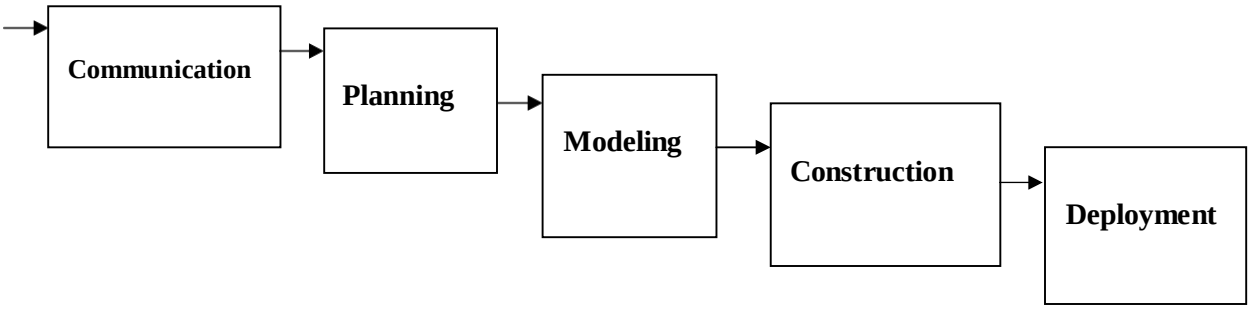
- Build self-directed teams
- Motivate the teams
- Acceptance of CMM level 5 behavior as normal to accelerate software process improvement
- Provide improvement guidance to high maturity organization

PROCESS MODELS

- Help in the software development
- Guide the software team through a set of framework activities
- Process Models may be linear, incremental or evolutionary

THE WATERFALL MODEL

- Used when requirements are well understood in the beginning
- Also called classic life cycle
- A systematic, sequential approach to Software development
- Begins with customer specification of Requirements and progresses through planning, modeling, construction and deployment.



This Model suggests a systematic, sequential approach to SW development that begins at the system level and progresses through analysis, design, code and testing

PROBLEMS IN WATERFALLMODEL

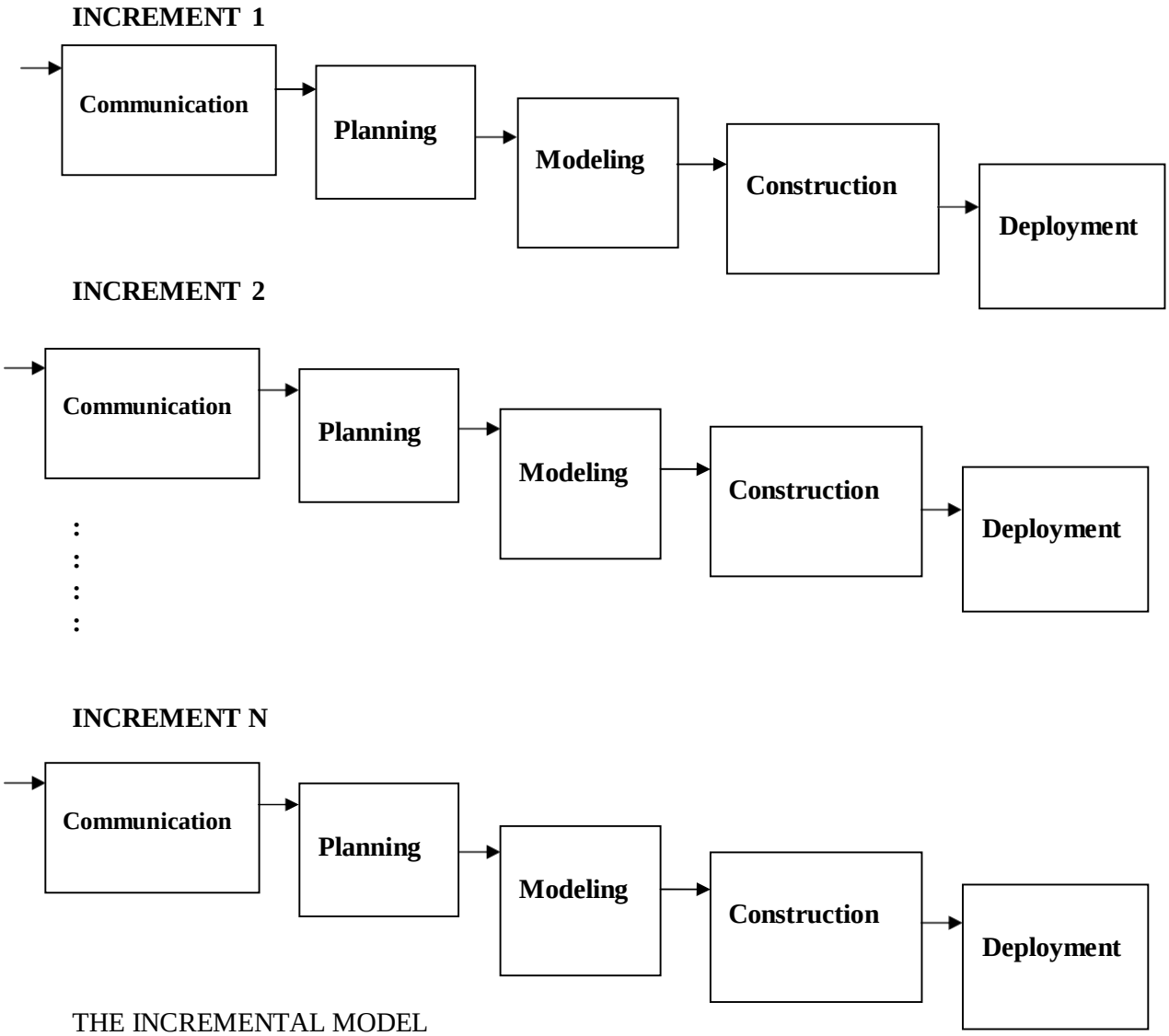
- Real projects rarely follow the sequential flow since they are always iterative
- The model requires requirements to be explicitly spelled out in the beginning, which is often difficult
- A working model is not available until late in the project time plan

THE INCREMENTAL PROCESS MODEL

- Linear sequential model is not suited for projects which are iterative in nature
- Incremental model suits such projects
- Used when initial requirements are reasonably well-defined and compelling need to provide limited functionality quickly
- Functionality expanded further in later releases
- Software is developed in increments

The Incremental Model

- Communication
- Planning
- Modeling
- Construction
- Deployment



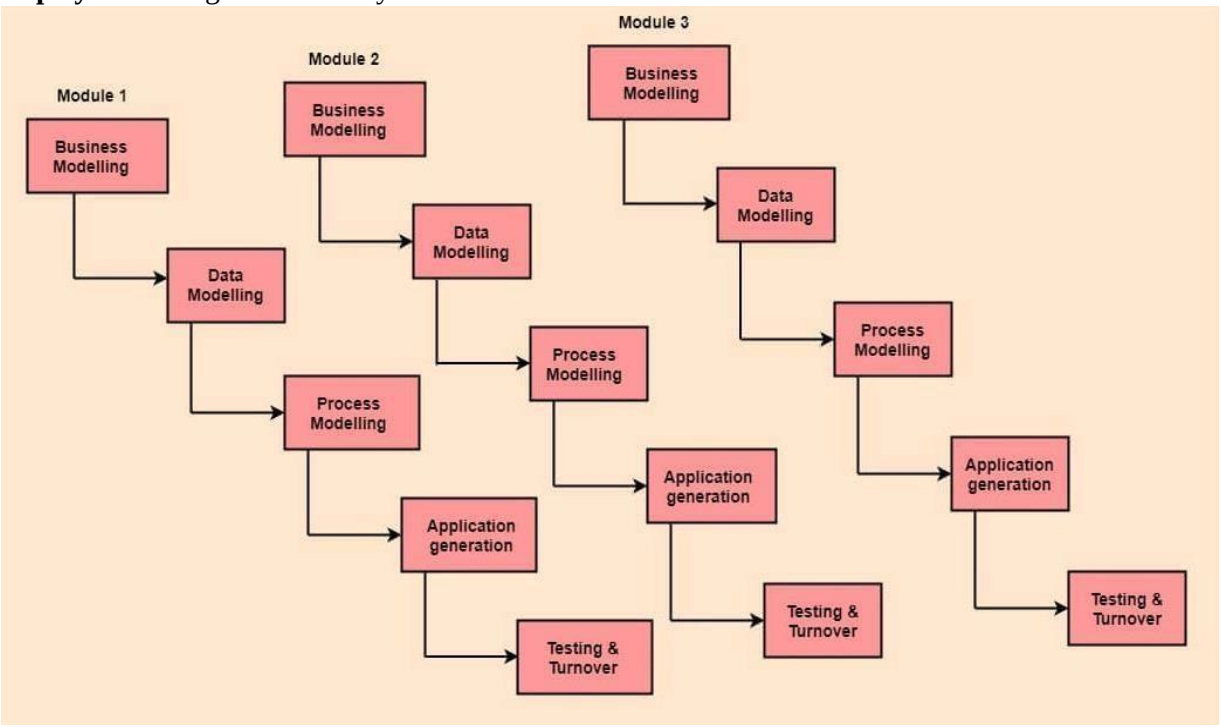
THE INCREMENTAL MODEL

- Software releases in increments
- 1st increment constitutes Core product
- Basic requirements are addressed
- Core product undergoes detailed evaluation by the customer
- As a result, plan is developed for the next increment. Plan addresses the modification of core product to better meet the needs of customer
- Process is repeated until the complete product is produced

THE RAD (Rapid Application Development) MODEL

- An incremental software process model
- Having a short development cycle
- High-speed adoption of the waterfall model using a component based construction approach
- Creates a fully functional system within a very short span time of 60 to 90 days

The RAD Model consists of the following phases:
Communication Planning Construction
Component reuses automatic code generation testing
Modeling
Business modeling Data modeling Process modeling
Deployment integration delivery feedback



THE RAD MODEL

- Multiple software teams work in parallel on different functions
- Modeling encompasses three major phases: Business modeling, Data modeling and process modeling
- Construction uses reusable components, automatic code generation and testing

Problems in RAD

- Requires a number of RAD teams
- Requires commitment from both developer and customer for rapid-fire completion of activities
- Requires modularity
- Not suited when technical risks are high

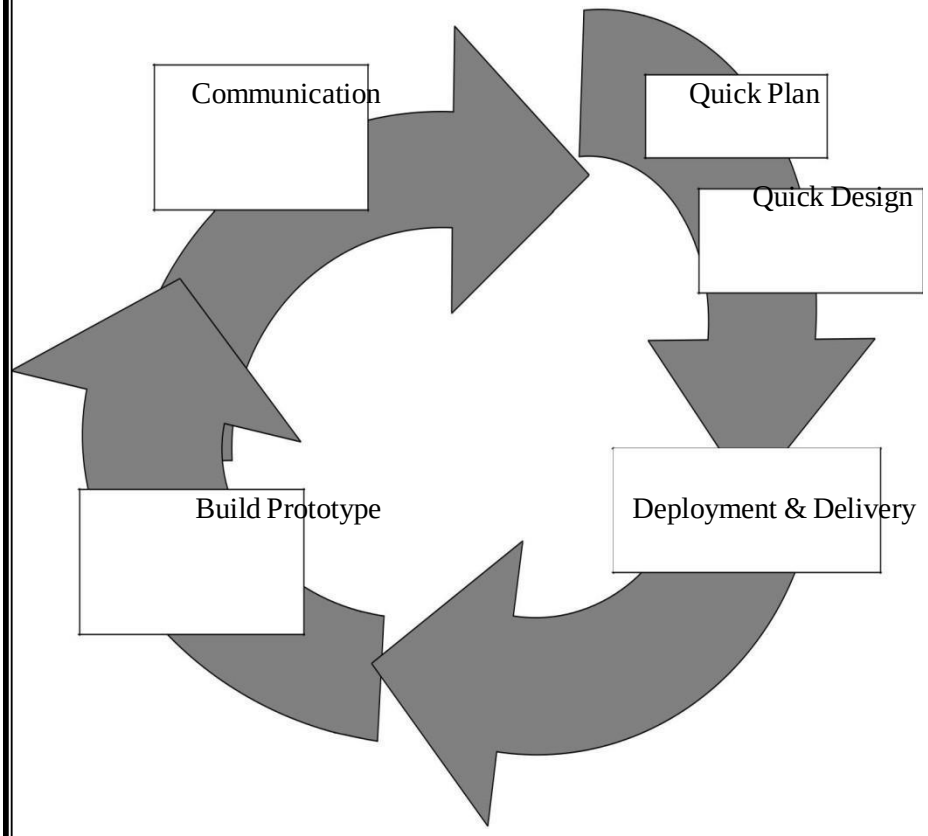
EVOLUTIONARY PROCESSMODEL

- Software evolves over a period of time
- Business and product requirements often change as development proceeds making a straight-line path to an end product unrealistic
- Evolutionary models are iterative and as such are applicable to modern day applications

Types of evolutionary models
– Prototyping
– Spiral model
– Concurrent development model

PROTOTYPING

- Mock up or model(throw away version) of a software product
- Used when customer defines a set of objective but does not identify input, output, or processing requirements
- Developer is not sure of:
– efficiency of an algorithm
adaptability of an operating system
– human/machine interaction



STEPS IN PROTOTYPING
- Begins with requirement gathering
- Identify whatever requirements are known
- Outline areas where further definition is mandatory
- A quick design occur
- Quick design leads to the construction of prototype
- Prototype is evaluated by the customer

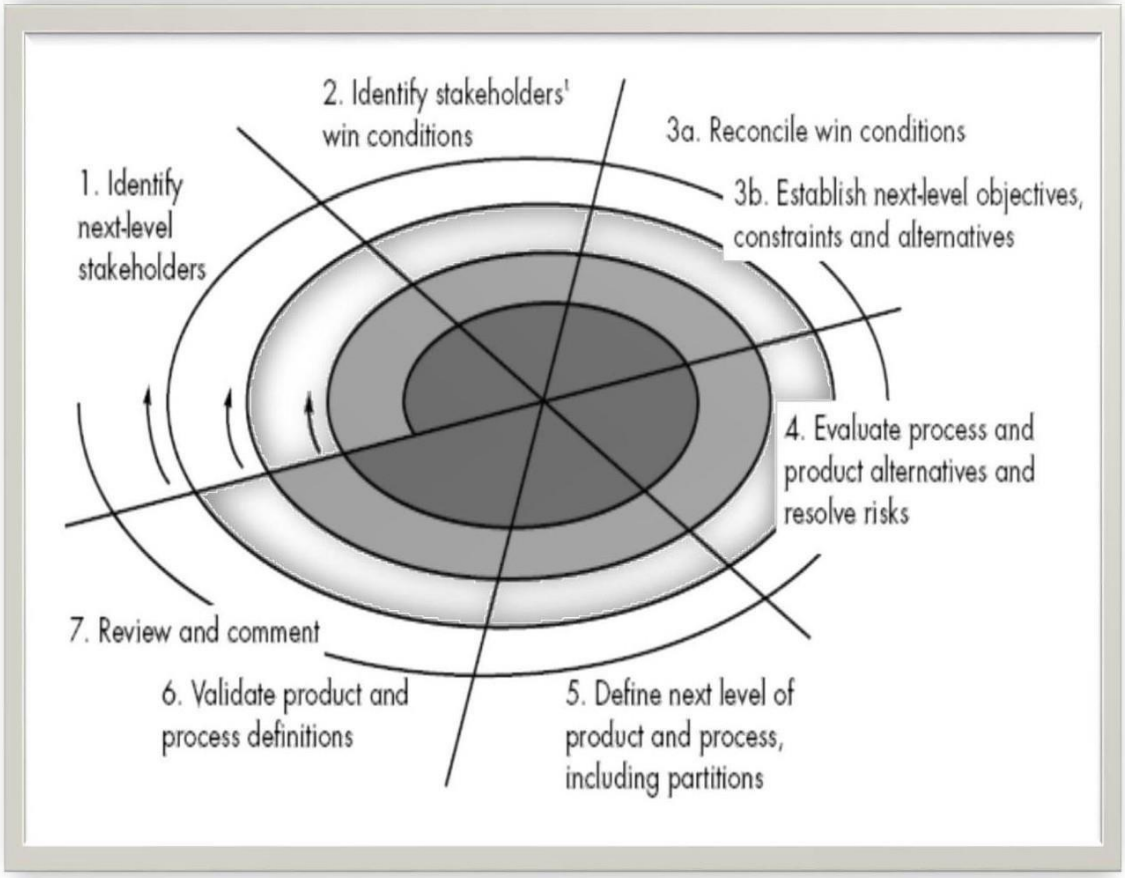
- Requirements are refined
- Prototype is turned to satisfy the needs of customer

LIMITATIONS OF PROTOTYPING

- In a rush to get it working, overall software quality or long term maintainability are generally overlooked
- Use of inappropriate OS or PL
- Use of inefficient algorithm

THE SPIRAL MODEL

An evolutionary model which combines the best feature of the classical life cycle and the iterative nature of prototype model. Include new element : Risk element. Starts in middle and continually visits the basic tasks of communication, planning, modeling, construction and deployment



THE SPIRAL MODEL

- Realistic approach to the development of large scale system and software
- Software evolves as process progresses
- Better understanding between developer and customer
- The first circuit might result in the development of a product specification

- Subsequent circuits develop a prototype
- And sophisticated version of software

THE CONCURRENT DEVELOPMENT MODEL

- Also called concurrent engineering
- Constitutes a series of framework activities, software engineering action, tasks and their associated states
- All activities exist concurrently but reside in different states
- Applicable to all types of software development
- Event generated at one point in the process trigger transitions among the states

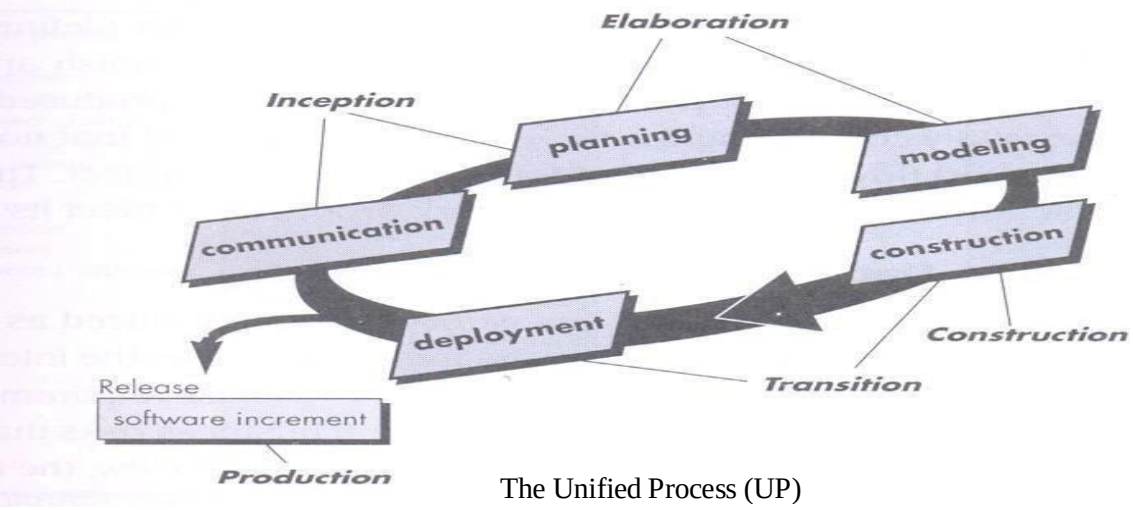
A FINAL COMMENT ON EVOLUTIONARY PROCESS
- Difficult in project planning
- Speed of evolution is not known

Does not focus on flexibility and extensibility (more emphasis on high quality)
- Requirement is balance between high quality and flexibility and extensibility

THE UNIFIED PROCESS

Evolved by Rumbaugh, Booch, Jacobson. Combines the best features their OO models. Adopts additional features proposed by other experts. Resulted in Unified Modeling Language (UML). Unified process developed Rumbaugh and Booch. A framework for Object-Oriented Software Engineering using UML

PHASES OF UNIFIED PROCESS
- INCEPTION PHASE
- ELABORATION PHASE
- CONSTRUCTION PHASE
- TRANSITION PHASE



The Unified Process (UP)

UNIFIED PROCESS WORK PRODUCTS

Tasks which are required to be completed during different phases

1. Inception Phase

*Vision document
*Initial Use-Case model
*Initial Risk assessment
*Project Plan

2. Elaboration Phase

*Use-Case model
*Analysis model
*Software Architecture description
*Preliminary design model
*Preliminary model

3. Construction Phase

*Design model
*System components
*Test plan and procedure
*Test cases
*Manual

4. Transition Phase

*Delivered software increment
*Beta test results
*General user feedback

Agility and Agile Process model

The meaning of Agile is swift or versatile."**Agile process model**" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance

Fig. Agile Model

Phases of Agile model:

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

1. **Requirements gathering:** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

2. **Design the requirements:** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

3. **Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

4. **Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

5. **Deployment:** In this phase, the team issues a product for the user's work environment.

6. Feedback: After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

Advantages:

1. Frequent Delivery
2. Face-to-Face Communication with clients.
3. Efficient design and fulfils the business requirement.
4. Anytime changes are acceptable.
5. It reduces total development time.

Disadvantages:

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.

2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

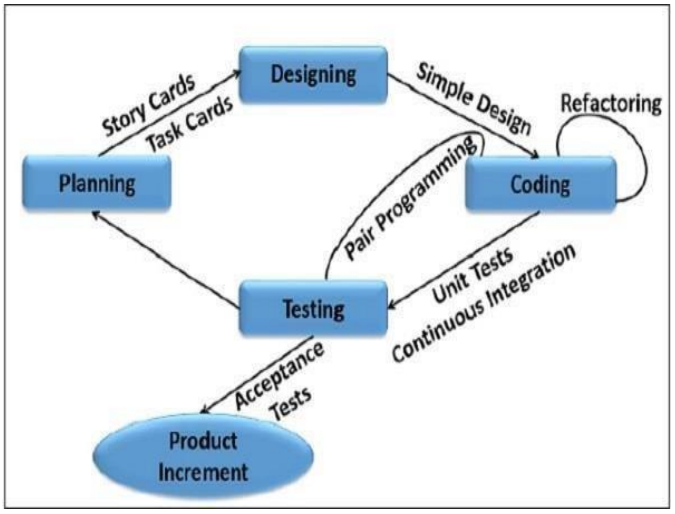
Extreme Programming

XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop software.

Extreme Programming (XP) was conceived and developed to address the specific needs of software development by small teams in the face of vague and changing requirements.

Extreme Programming is one of the Agile software development methodologies. It provides values and principles to guide the team behavior. The team is expected to self-organize. Extreme Programming provides specific core practices where −

- Each practice is simple and self-complete.
- Combination of practices produces more complex and emergent behaviour.

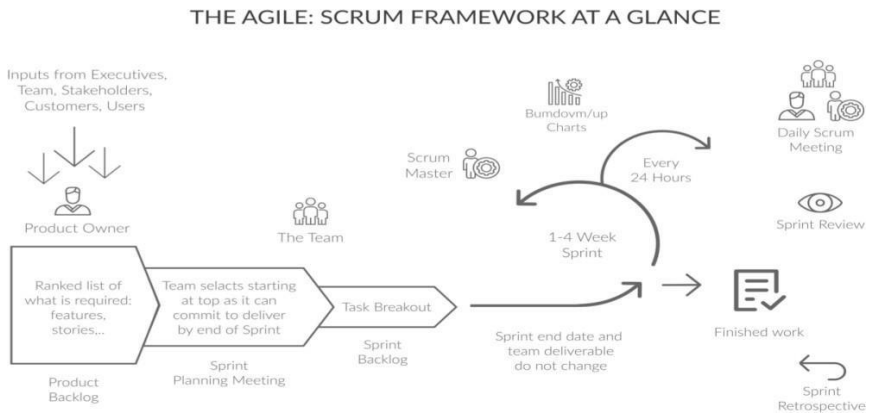


Other process models of Agile Development and Tools

- Crystal
- Scrum

Scrum

Scrum is aimed at sustaining strong collaboration between people working on complex products, and details are being changed or added. It is based upon the systematic interactions between the three major roles: Scrum Master, Product Owner, and the Team.



- **Scrum Master** is a central figure within a project. His principal responsibility is to eliminate all the obstacles that might prevent the team from working efficiently.
- **Product Owner**, usually a customer or other stakeholder, is actively involved throughout the project, conveying the global vision of the product and providing timely feedback on the job done after every sprint.
- **Scrum Team** is a cross-functional and self-organizing group of people that is responsible for the product implementation. It should consist of up to 7 team members, in order to stay flexible and productive.

Crystal

Crystal is an agile methodology for software development. It places focus on people over processes, to empower teams to find their own solutions for each project rather than being constricted with rigid methodologies.
Crystal methods focus on:-

People involved
Interaction between the teams
Community
Skills of people involved
Their Talents
Communication between all the teams