CHAPTER: 1

SOFTWARE DEVELOPMENT PROCESS

(12 Marks)

1.1 Software, Software Engineering as layered approach and its Characteristics, Types of software.
1.2 Software development framework.
1.3 Software Process Framework, Process models, Specialized Process Models.
1.4 Agile Software development: Agile process & its importance, Extreme programming, Adaptive Software Development, Scrum, Dynamic Systems Development method (DSDM), Crystal.
1.5 Selection criteria for software process model.

1.1 Software & Its Characteristics

- Computer software is a product that design & built by software engineers.
- It is a collection of programs, documentation & operating procedures.
- ***i. e. software=program +documentation + operating procedures.***
- **Software encompasses:**
  (1) Instructions (computer programs) that when executed provide desired features, function, and performance.
  (2) Data structures that enable the programs to adequately store and manipulate information &
  (3) Documentation that describes the operation and use of the programs.

1.1.1 Features or Characteristics

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

- Although some similarities exist between software development and hardware manufacture, the two activities are fundamentally different.
- In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems that are non-existent (or easily corrected) for software.
- Both activities are dependent on people, but the relationship between people applied and work accomplished is entirely different.
- Software costs are concentrated in engineering. This means that software projects cannot be managed as if they were manufacturing projects.

2. Software doesn't "wear out."

- Figure 1.1 depicts failure rate as a function of time for hardware.

- The relationship, often called the "bathtub curve," indicates that hardware exhibits relatively high failure rates early in its life.
- As time passes, however, the failure rate rises again as hardware components suffer from the cumulative effects of dust, vibration, abuse, temperature extremes, and many other environmental maladies. Stated simply, the hardware begins to *wear out.*

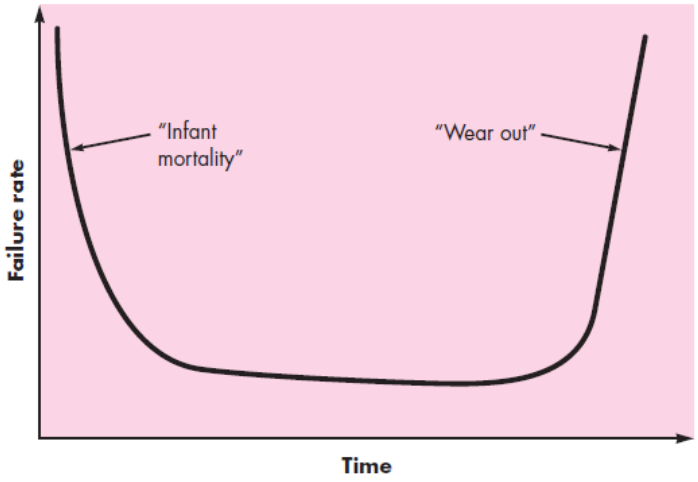


Fig. 1.1 Failure curve for hardware

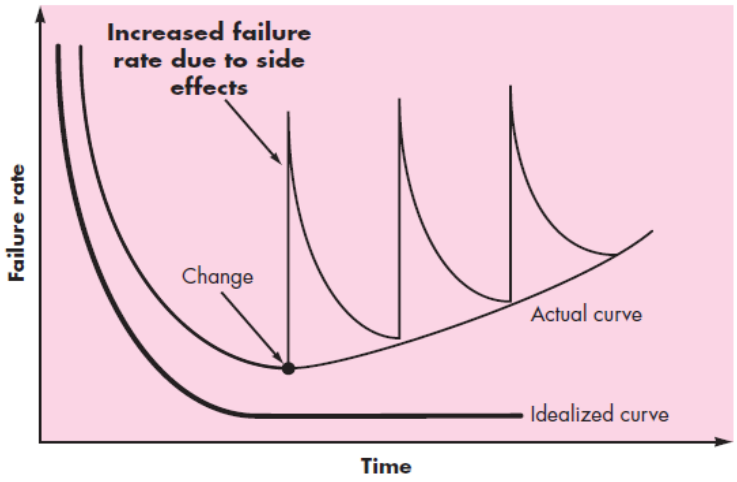


Fig. 1.2 Failure curves for software

- The failure rate curve for software should take the form of the "idealized curve" shown in Figure 1.2.
- Undiscovered defects will cause high failure rates early in the life of a program. However, these are corrected and the curve flattens as shown.
- The idealized curve is a gross oversimplification of actual failure models for software.
- However, the implication is clear—software doesn't wear out. But it does *deteriorate!*
- This contradiction can best be explained by considering the "actual curve" shown in Figure.
- During its life, software will undergo change (maintenance). As changes are made, it is likely that some new defects will be introduced, causing the failure rate curve to spike as shown in Figure.
- Before the curve can return to the original steady-state failure rate, another change is requested, causing the curve to spike again. Slowly, the minimum failure rate level begins to rise—the software is deteriorating due to change.

3. Software is not susceptible (influenced) to the environmental melodies.

- As time passes, however, the failure rate rises again as hardware components suffer from the cumulative effects of dust, vibration, abuse, temperature extremes, and many other environmental maladies.
- Software is not susceptible to the environmental maladies that cause hardware to wear out.

4. Most Software is Custom-Built, rather than being assembled from old existing components.

- The reusable components have been created so that the engineer can concentrate on the truly innovative elements of a design, that is, the parts of the design that represent something new.
- In the software world, it is something that has only begun to be achieved on a broad scale. A software component should be designed and implemented so that it can be reused in many different programs
- A software component should be designed and implemented so that it can be reused in many different programs. Modern reusable components encapsulate both data and the processing that is applied to the data, enabling the software engineer to create new applications from reusable parts.
- **For example, today's interactive user interfaces are built with reusable components that enable the creation of graphics windows, pull-down menus, and a wide variety of interaction mechanisms.**

1.1.2 Types of software or Changing Nature of Software

- **System software: -** It control & manages computer hardware, so that user can perform some specific application software. (e.g. OS)
- **Application software: -** Designed to satisfy a particular need of a particular environment.(e.g. student management system).
- **Engineering/scientific software: -** Modern applications within the Engg./Scientific area are moving away from conventional numerical algorithms.(e.g. system simulation, MATLAB etc.)
- **Embedded software: -** S/W resides within a product or system.(e.g. Washing machine, keypad of microwave etc.)
- **Product-line software** (e.g., inventory control, word processing, multimedia)
- **Web applications:-** acts as an interface between the user & internet((e.g. Browser, E-commerce)
- **Artificial intelligence software: -** uses non numerical algorithms,(e.g. robotics, expert systems, image & voice recognitions etc.)
- **Open source** (operating systems, databases, development environments)
- **Real time software:-** it monitors/ analyses/ control real world events as they occur. It must respond within strict time constraints.
- **Personal computer software** :-(word processor, multimedia, games, DBMS, etc.)

1.1.3 Software Engineering as layered approach

- **Software Engineering Definition "** ***Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software "***

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- **Need of S/W Engg**
- Software product are complicated they need scientific & Engg. Approach to develop.
- Software development is expensive so proper measures are required.
- Cost & time considerations are other factors.
- Ability to produce new applications to meet the market needs.
- Building complex & critical s/w system in a timely manner & with high quality.
- ***For above problems the term software Engg. Is formed. It has the objective of solving these problems by producing good quality, maintainable software or time, within budget.***

1.1.4 Software Engineering a Layered Technology

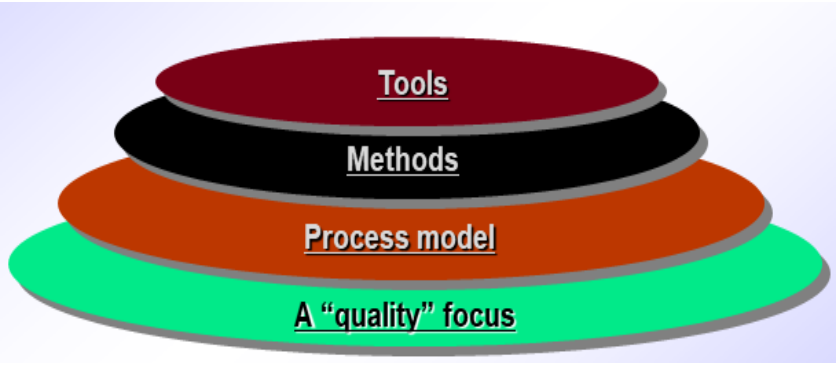


Fig. 1.3 Software Engg a Layered Technology

- Any engineering approach must rest on organizational commitment to **quality** which fosters a continuous process improvement culture.
- **Process** layer as the foundation defines a framework with activities for effective delivery of software engineering technology.
    - 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 doing the work to pick and choose the **appropriate set of work actions** and tasks.
    - Purpose of process is to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.
- **Method** provides technical how-to be for building software. It encompasses many tasks including communication, requirement analysis, design modeling, program construction, testing and support.
- **Tools** provide automated or semi-automated support for the process and methods.

1.2 Software development framework.

Software Process Framework

Q. Explain Process framework with a suitable diagram. W-2019

- A process framework establishes the foundation for a complete software process by identifying a small number of framework activities that are applicable to all software projects.
- In addition, the process framework encompasses a set of umbrella activities that are applicable across the entire software process.

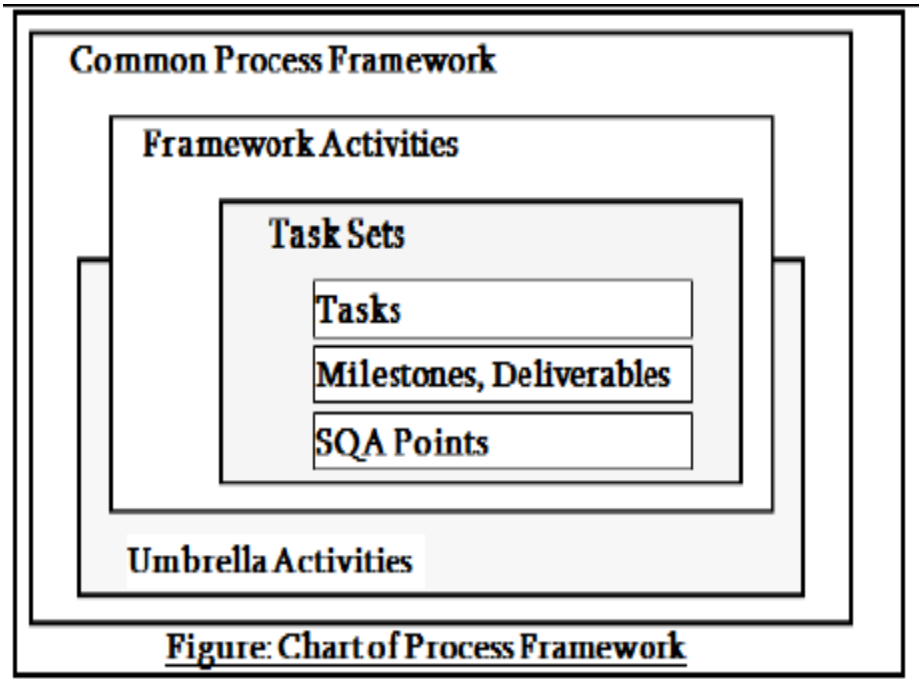


Fig. Process Framework

Basic framework activities:

1. Communication:

- This framework activity involves heavy communication & collaboration with the customer (and the stakeholders) and encompasses requirements gathering and other related activities.

2. Planning:

- This activity establishes a plan for the software engineering work that follows.
- It describes the technical tasks to be conducted; the risks are analyzed.
- Project tracking should be done.
- Deadline is fixed.

3. Modeling:

- This activity encompasses the creation of models that allow the developer & the customer to better understand software requirements & the design that will achieve those requirements.

4. Construction:

- This activity combines code generation and the testing that is required uncovering errors in the code.

5. Deployment:

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

Umbrella Activities

1. Software project tracking and control: Assess progress against the plan and take actions to maintain the schedule.
2. Software quality assurance (SQA): Defines and conduct activities to ensure quality.
3. Software configuration management: Manage the effects of change throughout the software process.
4. Risk management: Assesses risks that may affect the outcome and quality.
5. Reusability management: Process of managing reusable part of product which is once developed can be used many times.
6. Technical reviews: Assesses work products to uncover and remove errors before going to the next activity.
7. Measurement: Define and collects process, project, and product measures to ensure stakeholder's needs are met.
8. Work product preparation and production: Create work products such as models, documents, logs, forms and lists.

1.3 Process models:-

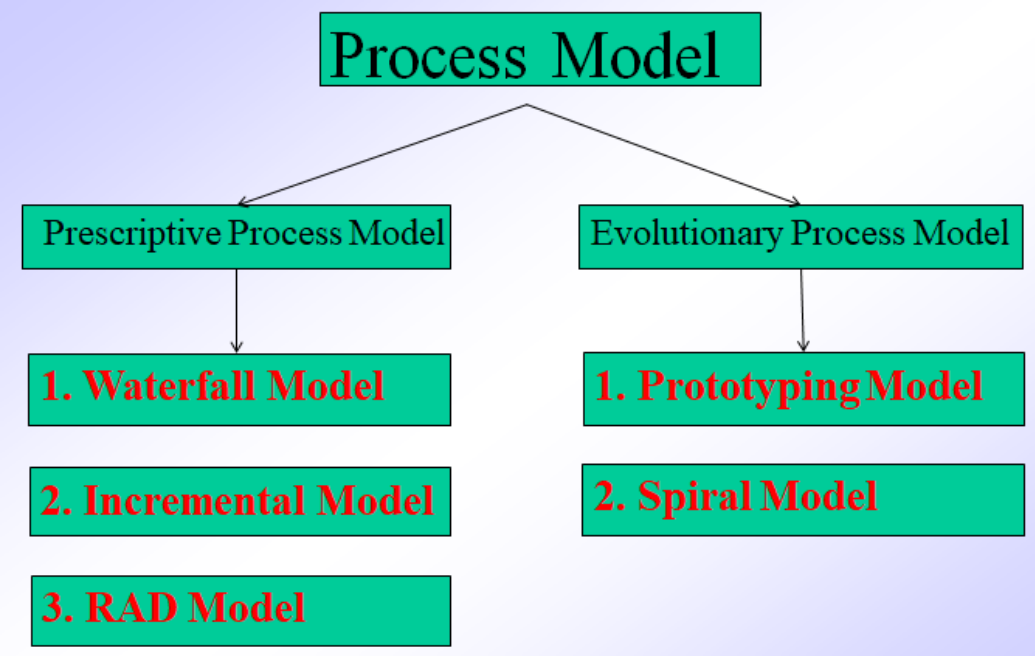


Fig. Hierarchy of process model

Prescriptive Process Model

- Defines a distinct set of activities, actions, tasks, milestones, and work products that are required to engineer high-quality software.
- The activities may be linear, incremental, or evolutionary.

1. Waterfall Model:

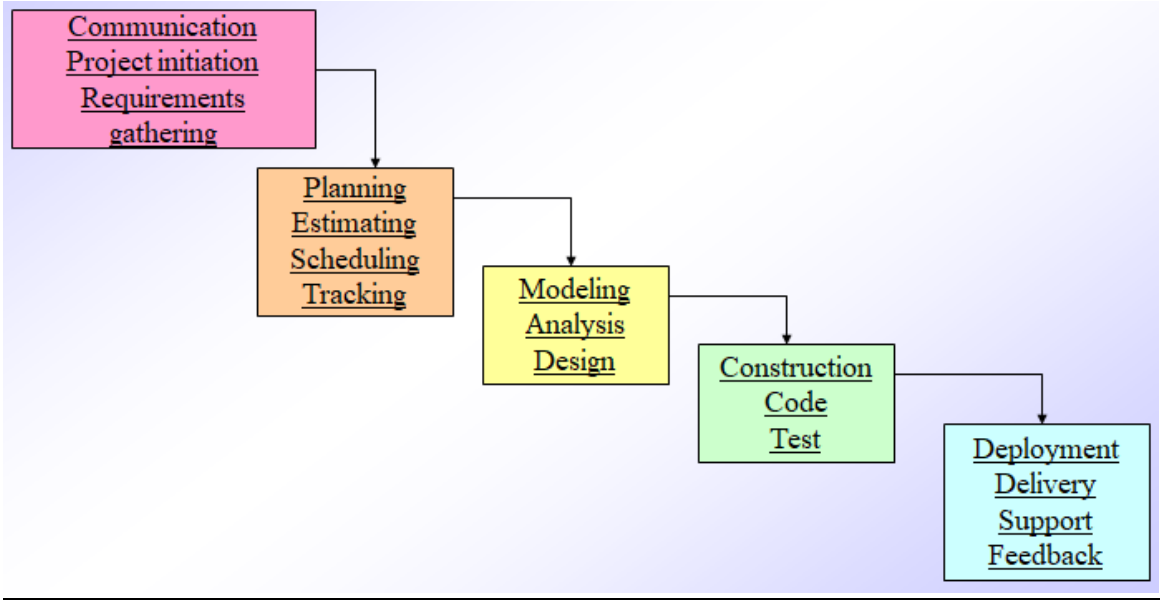


Fig. Waterfall Model

- The waterfall model is a traditional method, sometimes called the classic life cycle.
- This is one of the initial models. As the figure implies stages are cascaded and shall be developed one after the other.
- It suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through, communication, planning, modeling construction and deployment.
- **Phases of Waterfall Model:-**
  - **Communication**
    - Involves communication among the customer and other stake holders; encompasses requirements gathering
  - **Planning**
    - Establishes a plan for software engineering work; addresses technical tasks, resources, work products, and work schedule.
  - **Modelling (Analysis, Design)**
    - Encompasses the creation of models to better under the requirements and the design.
  - **Construction (Code, Test)**

  - Combines code generation and testing to uncover errors.
- **Deployment**
  - Involves delivery of software to the customer for evaluation and feedback.

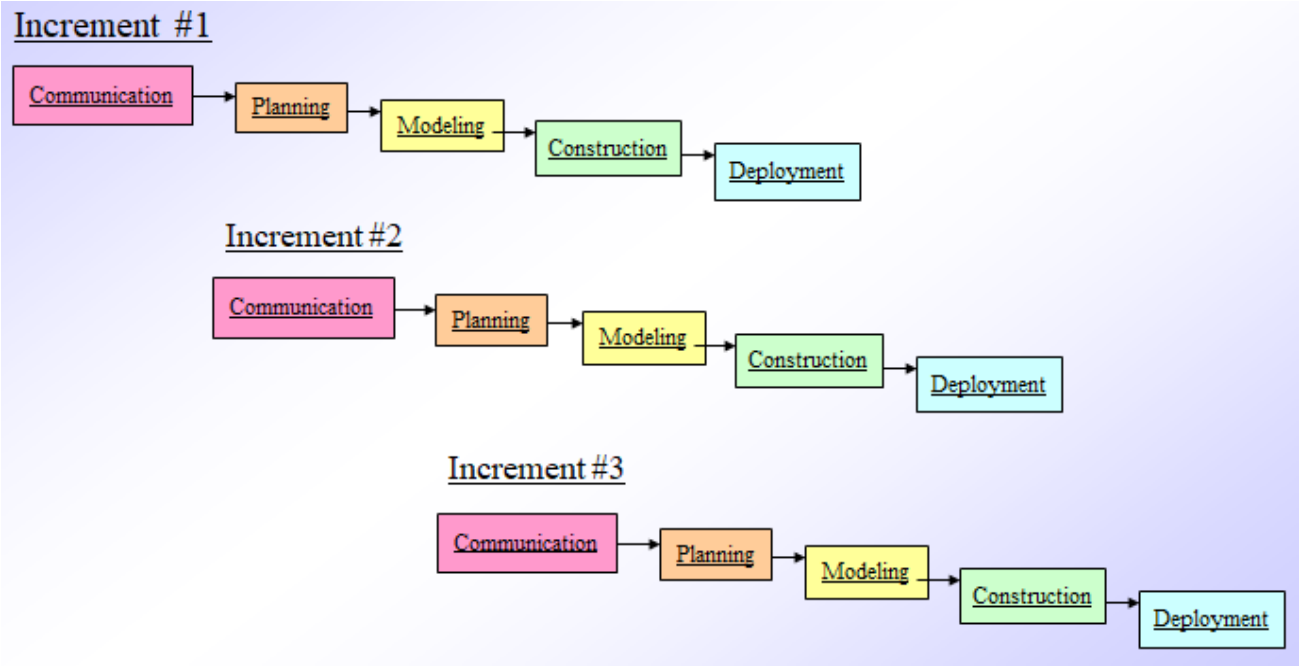
Advantages:-

- Oldest software lifecycle model and best understood by upper management
- Used when requirements are well understood and risk is low.
- Work flow is in a linear (i.e., sequential) fashion.
- Used often with well-defined adaptations or enhancements to current software.
- In waterfall model progress of system is measureable.
- After every step of software coding, testing is done to check the correct running of the code.

Disadvantage:-

- The model in linear, rigid & monolithic.
- Does not involve risk management.
- A requirement needs to be specified before the development proceeds.
- Doesn't support iteration, so changes can cause confusion
- Difficult for customers to state all requirements explicitly and up front

2. Incremental Model



- Used when requirements are well understood.
- Multiple independent deliveries are identified.
- Work flow is in a linear (i.e., sequential) fashion within an increment and is staggered between increments.
- Iterative in nature; focuses on an operational product with each increment.

- Provides a needed set of functionality sooner while delivering optional components later
- **Ex. Word processing software developed.**
  1. Basic file operation, editing, formatting etc. -----function in 1st increment.
  2. More efficient editing & formatting may be done in 2nd increment.
  3. May synonyms & grammar related checking will be done in 3rd increment.

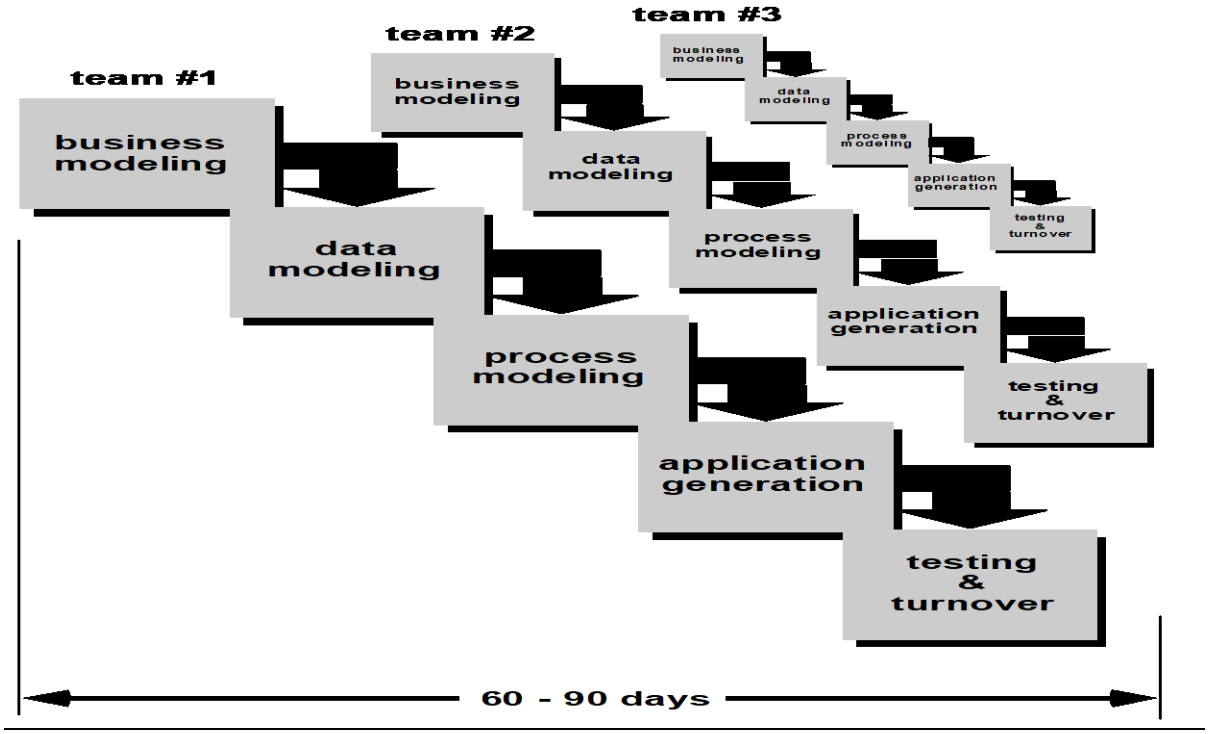
Advantages:-

- It combines iterative nature of prototyping model & linear nature of linear sequential model.
- Number of people required is less.
- Easy to add quality.
- Deadlines can be managed in an effective manner.

Disadvantages:-

- Integration testing is difficult to do.
- Assessment of project risks & its resolution is not an easy work.
- Each quality developed requires testing.
- Reusability of codes among the modules is minimum.

3. RAD (Rapid Application Development) Model



- **Business Modeling.** The information flow among business functions is modeled in a way that answers the following questions: What information drives the business process? What information is generated? Who generate it? Where does the information go? Who process it?

- **Data Modeling.** The information flow defined as part of the business modeling phase is refined into a set of data objects that are needed to support the business.
- **Process Modeling.** The data objects defined in the data modeling phase are transformed to achieve the information flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.
- **Application Generation.** The RAD process works to reuse existing program components (when possible) or create reusable components (when necessary). **In all the cases, automated tools are used to facilitate construction of the software.**
- **Testing and Turnover.** Since the RAD process emphasizes reuse, many of the program components have already been tested. This reduces overall testing time.

Advantages of RAD:-

- Continues customer & developer interaction there.
- Flexible & adaptable to changes.
- RAD generally incorporates short development cycles.
- It works with great speed.

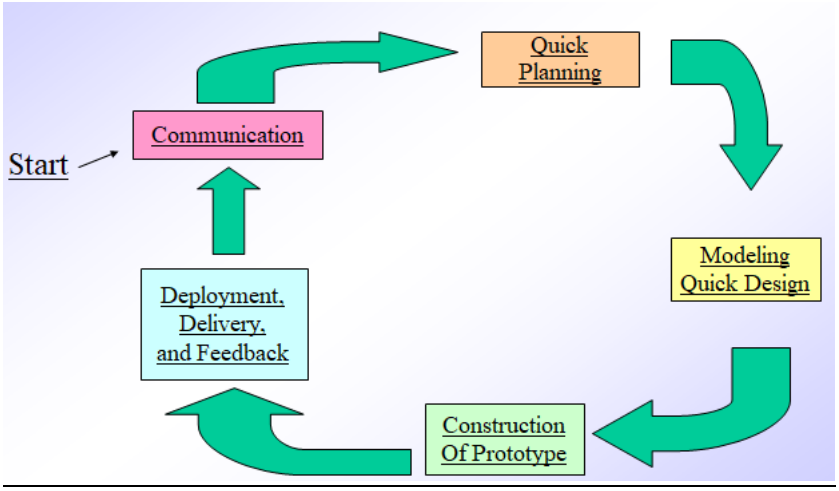
Disadvantages of RAD:-

- Useful for only large projects.
- RAD needs enough human resources to create the required no. of teams.
- If developer & customer are not committed to the rapid model, the RAD project fails.
- When technical risks are high, RAD may not be a suitable option.

Evolutionary Models

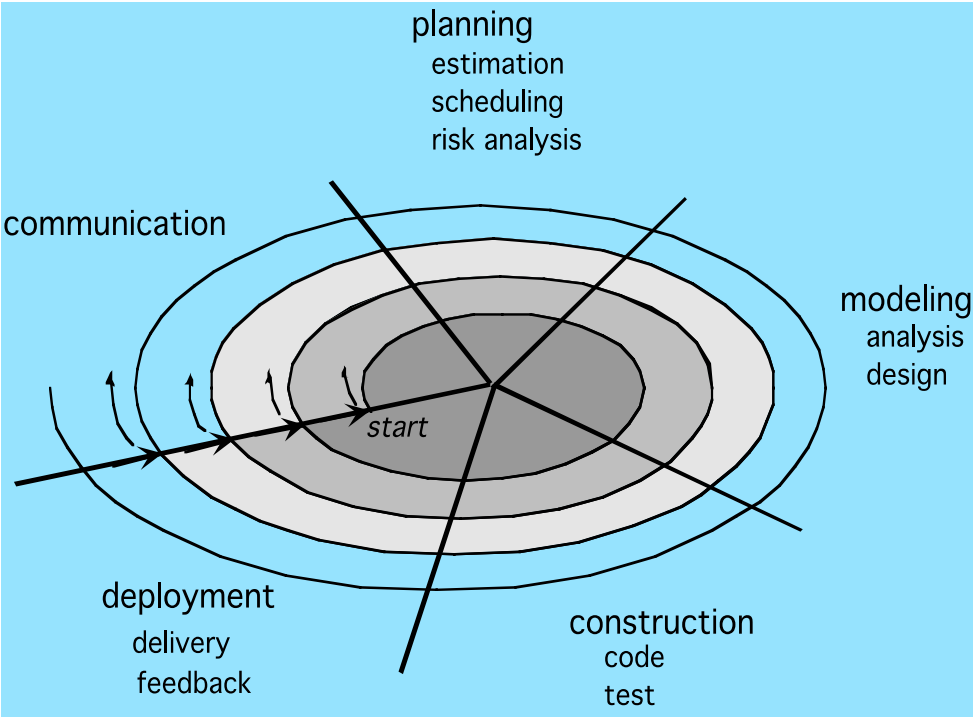
- It is iterative or repetitive that enables you to develop increasingly more complete version of the software.
- Two types are introduced, namely Prototyping and Spiral models.

1. Prototyping Model



- **When to use?:** Customer defines a set of general objectives but does not identify detailed requirements for functions and features
- Used when requirements are not well understood.
- Follows an evolutionary and iterative approach.
- Serves as a mechanism for identifying software requirements.
- Focuses on those aspects of the software that are visible to the customer/user.
- Feedback is used to refine the prototype.
- **What step:**
- **Communication:** - Begins with communication by meeting with stakeholders to define the objective, identify whatever requirements are known, and outline areas where further definition is mandatory.
- **A quick plan** for prototyping and modeling (quick design) occur.
- **Quick design** focuses on a representation of those aspects the software that will be visible to end users. (Interface and output).
- Design leads to the **construction of a prototype** which will be deployed and evaluated. Stakeholder's comments will be used to refine requirements for the s/w.
- It is the iterative or repetitive till the customer satisfaction.
- Both stakeholders and software engineers like the prototyping paradigm. Users get a feel for the actual system, and developers get to build something immediately.

2. The Spiral Model



- Invented by Dr. Barry Boehm in 1988 while working at TRW
- Follows an evolutionary approach
- **Used when requirements are not well understood and risks are high.**
- Inner spirals focus on identifying software requirements and project risks; may also incorporate prototyping.
- Outer spirals take on a classical waterfall approach after requirements have been defined, but permit iterative growth of the software.
- Operates as a risk-driven model…a go/no-go decision occurs after each complete spiral in order to react to risk determinations.
- Requires considerable expertise in risk assessment.
- Serves as a realistic model for large-scale software development.
- **It couples the iterative nature of prototyping with the** controlled and systematic aspects of the waterfall model and is a risk-driven process model generator that is used to guide multi stakeholder concurrent engineering of software intensive systems.
- The **first circuit in the clockwise direction might** result in the product specification; subsequent passes around the spiral might be used to develop a prototype and then progressively more sophisticated versions of the software.
- Each pass results in adjustments to the project plan. Cost and schedule are adjusted based on feedback. Also, the number of iterations will be adjusted by project manager.
- Good to develop large-scale system as software evolves as the process progresses and risk should be understood and properly reacted to. Prototyping is used to reduce risk.

Q. Compare Prescriptive process model and agile process model

Prescriptive Model	Agile Process Model
It is set of predefined steps that are also known as classical model	Agile model are evolved over the period of time.
Focuses on every activity equally	Focuses on the deliverable that fulfills user's requirements.
Takes more time that may not be realistic	Takes much less time
Reverse Engg. is difficult	Reverse Engg. is easy.
Software configuration managements is difficult	Software configuration managements is manageable.
Results are produced with delay	Quick results are produced.
It is Process oriented.	It is people oriented.
It follows Life cycle model (waterfall, spiral) development model.	It follows Iterative and Incremental development model.

Documentation required is to be comprehensive and constant.	Documentation required is to be minimal and evolving.
Predictive planning is required.	Adaptive planning is required.
Customer's role is important.	Customer's role is critical.
Formal communication is required.	Informal communication is required.

1.4 Agile Software development: Agile process & its importance,

Q. What is "Agility"? (Ability to move quickly and easily.)

- Agile software engineering combines a philosophy and a set of development guidelines.
- It is the unending process, which always accepts specific requirement of the product from the customers or end user then it checks whether the requirements are growth oriented or not? If it is growth oriented then it aggressively changes the existing system.
- The philosophy encourages customer satisfaction and early incremental delivery of software; small, highly motivated project teams; informal methods; minimal software engineering work products; and overall development simplicity.

Agile Software Development (ASD)

- Agile software engineering combines a philosophy and a set of development guidelines.
- It focuses on rapid development of software product by considering current market requirement & time limit.
- Today's market is rapidly changing & unpredictable too.
- Root of agile software development is in the reality of today's markets.
- If we compare prescriptive model with ASD refined ASD is very much useful & **practically applicable.**
- ASD focuses on **face to face or interactive** processes then documentation.
- ASD **saves the man power, cost, documentation & most importantly time**.

Features of ASD approach

- No boss & sub ordinate relationships work in the contest of work.
- Team members are to take work related decision
- This model takes much less time for development of product as compared to other model.
- Continuous learning, knowledge sharing & knowledge creation.
- It enforces more comfortable working environments
- Working culture is co-operative.
- Allows quality product to be developed.
- Works as per current requirement of the market.

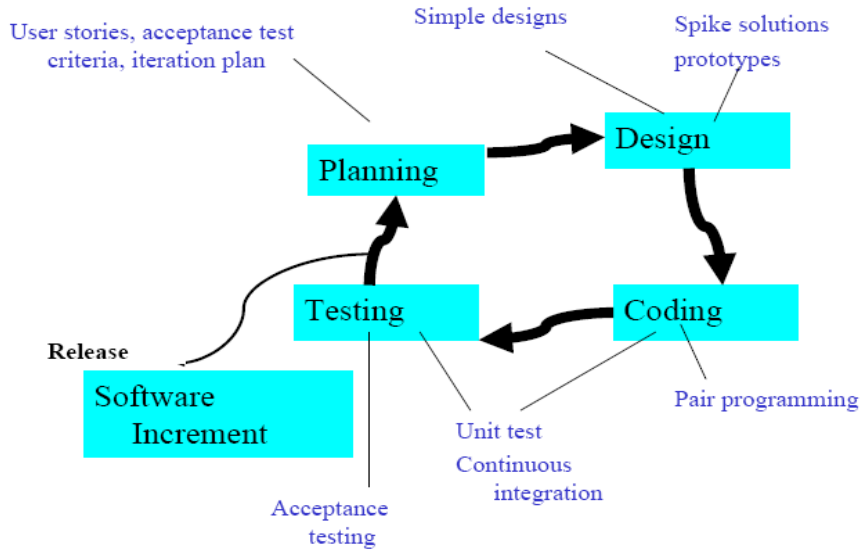
Agile method or Process Model

- **Most popular agile methodological include:**
  1. **Extreme Programming(XP)**

2. **ASD (Adaptive Software Development)**
3. **Scrum**
4. **Dynamic Systems Development method (DSDM)**
5. **Crystal**

1.4.1 Extreme programming

- Extreme programming 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 behavior.
- Extreme Programming is based on the following values-
  - **Communication:** Everyone on a team works jointly at every stage of the project.
  - **Simplicity:** Developers strive to write simple code bringing more value to a product, as it saves time and efforts.
  - **Feedback:** Team members deliver software frequently, get feedback about it, and improve a product according to the new requirements.
  - **Respect:** Every person assigned to a project contributes to a common goal.
  - **Courage:** Programmers objectively evaluate their own results without making excuses and are always ready to respond to changes.



- XP encompasses a set of rules and practices that occur within the context of four framework activities**: planning, design, coding and testing. (See above Fig.)**
- **Planning**
  - Begins with a set of stories (scenarios).
  - Each story written by the customer is assigned a value depending on its priority.
  - The members of the XP team assess each story and assigned a cost measured in development weeks.
  - If a story has more than three weeks to develop the customer is asked to split it.
  - New stories can add any time.
  - Customers and XP team work together to decide how to group stories for next increment.
  - The XP team then reconsiders all remaining releases and modifies its plan accordingly.
- **Design**
  - A simple design is preferred.
  - Design only considers the given stories.
  - Extra functionality discouraged.
  - Identify the object oriented classes that are relevant to the current system.
- **Coding**
  - XP recommends developing a series of unit tests for each of the story.
  - Once the code is complete, units should be unit tested.
  - **Pair programming –** two people work together at one computer.
- **Testing**
  - The unit test that has been created in the coding stage should be implemented using a framework that can be implemented.
  - Integration and validation can occur on a daily basis.
  - This provides the XP team with a continual indication of the progress and also raises flags early if things are going wrong.
  - Acceptance tests are derived from user stories that have been implemented as parts of the software release.

1.4.2 Adaptive Software Development

- A software development process that grew out of rapid application development work.
- It has been proposed by Jim Highsmith as a technique for building complex software and system.
- It focuses on human collaboration and team self-organization.

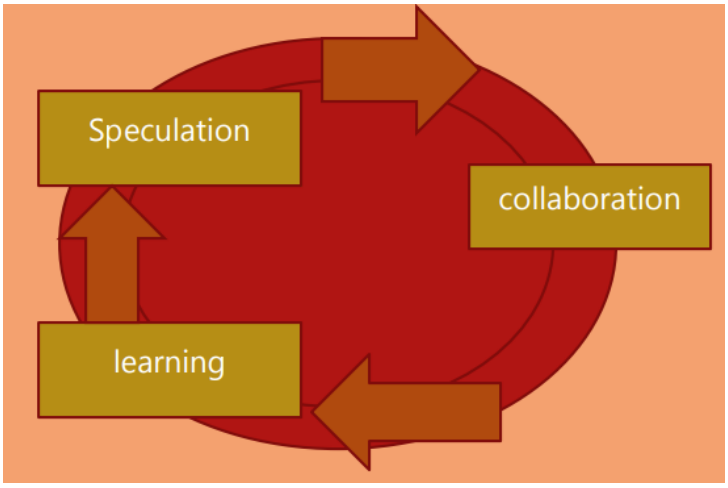


Fig. Life Cycle of Adaptive Software Development

Speculation

- During speculation, the project is initiated and adaptive cycle planning is conducted.
- When we speculate, it's not that we don't define a mission to the best of our ability.

Collaboration

- Encompasses communication and teamwork but also emphasizes individualism.
- If we can't predict, then we can't control in the traditional management sense.
- If we can't control, then a significant set of current management practices is no longer operable.

Learning

- Challenges all stakeholders.
- Examine their assumptions and use the results of each development cycle to learn the direction of the next.
- Customer focus groups, technical reviews, beta testing, and postmortems are all practices that expose results to scrutiny.

Advantages

- Software incremental adjustment.
- Rapid and complex software development

Disadvantages

- There is lack of emphasis on necessary designing and documentation.
- It is difficult to assess the effort required at the beginning of the software development life cycle.

1.4.3 Scrum (Definition) (The name is derived from an activity that occurs during a rugby match)

- Scrum is one of the Agile frameworks followed to complete challenging projects wherein there are dynamic changes in the requirements by using one or more cross-functional, self-organizing teams of about 7 +/- 2 people on each team.

- Scrum consists of 3 roles: product owner, ScrumMaster, and development team/engineering team.
- Scrum uses fixed-length iterations called sprints ranging from one Week four weeks (or 30 days long).
- Scrum teams should consist of 7 +/- 2 people.
- Sprint cannot exceed more than 30 days.
- Sprint is time bound.
- Scrum team aims to build a potentially shippable product by end of each sprint.
- Daily Scrum / Stand-up meeting should be only for 15 mins. and as the name goes, the team should be standing during entire meeting time.

1.4.4 Dynamic Systems Development method (DSDM)

- The Dynamic Systems Development Method (DSDM) is an agile framework that addresses the entire project lifecycle and its impact on the business.
- DSDM is an iterative approach to software development, and this framework explicitly states "any project must be aligned to clearly defined strategic goals and focus upon early deliver of real benefits to the business."
- The framework is built on four principles: feasibility and business study, functional model and prototype iteration, design and build iteration, and implementation.
- **Feasibility Study:**
  It establishes the essential business necessities and constraints related to the applying to be designed then assesses whether or not the application could be a viable candidate for the DSDM method.
- **Business Study:**
  It establishes the use and knowledge necessities that may permit the applying to supply business value.
- **Functional Model Iteration:**
  It produces a collection of progressive prototypes that demonstrate practicality for the client.
- **Design and Build Iteration:**
  It revisits prototypes designed throughout useful model iteration to make sure that everyone has been designed during a manner that may alter it to supply operational business price for finish users.
- **Implementation:**
  It places the newest code increment (an "operationalized" prototype) into the operational surroundings. It ought to be noted that:
  (a) The increment might not 100% complete or,
  (b) Changes are also requested because the increment is placed into place.

1.4.5 Crystal

- Alistair Cockburn, credited as one of the original popularizers of agile, developed the Crystal method for IBM in 1991.
- He decided to focus not on developing specific step-by-step development strategies that would work across the board for teams involved in any project, but instead to develop guidelines for team collaboration and communication.
- **Crystal Methods, which is a collection of agile software development approaches, focuses primarily on people and the interaction among them while they work on a software development project.**
- **The Crystal agile framework is built on two core beliefs:**
  - Teams can find ways on their own to improve and optimize their workflows.
  - Every project is unique and always changing, which is why that project's team is best suited to determine how it will tackle the work.
- Crystal focuses on six primary aspects: people, interaction, community, communication, skills, and talents.
- Process is considered secondary.
- The methods are very flexible and avoid rigid processes because of its human-powered or people-centric focus.
- It is also lightweight, without too much documentation, management or reporting.
- The weight of the methodology is determined by the project environment and team size.

  For example,
  - Clear - for teams of 8 or fewer people
  - Yellow - for teams of 10-20 people
  - Orange - for teams of 20-50 people
  - Red - for teams of 50-100 people

1.5 Selection Criteria for software process model

- The software process model framework is specific to the project. Thus, it is essential to select the software process model according to the software which is to be developed.
- The software project is considered efficient if the process model is selected according to the requirements.
- It is also essential to consider time and cost while choosing a process model.
- The basic characteristics required to select the process model are project type and associated risks, requirements of the project, and the users.
- Key features of selecting a process model are to understand the project in terms of size, complexity, funds available, and so on.

----------End of Chapter----------