CHAPTER - 1

Introduction to Software Engineering

1.1. Computer software

Computer software is the product that software professionals build and then support over the long term. A textbook description of software might take the following form: Software is:

1. Instructions (computer programs) that when executed provide desired features, function, and performance;
2. Data structures that enable the programs to adequately manipulate information, and
3. Descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.

Software has characteristics that are considerably different than those of hardware:

1. Software is developed or engineered; it is not manufactured in the classical sense.
2. Software doesn't wear out, but it does deteriorate.
3. Although the industry is moving toward component-based construction, most software continues to be custom built.

1.2 History of Software Engineering

- In the 1950's software development was de-emphasized, because it only contributed to about 20% of overall system cost. The hardware manufacturing company generally assign their software job to third party.
- Programmers moved from machine language, to assembly language, to high-level language. Also, the concern of organizational privacy increase and the hardware manufacturing company stated to develop software for their hardware product.
- In 1968, a NATO report coined the term "software engineering". It discuss about the group work for software engineering in more systematic way.
- Hardware became faster and cheaper, outpacing the ability of software to keep up
- By the 1980's the software cost of a system had risen to 80%, and many experts pronounced the field "in crisis" because the software development face the problem of not delivered on time, over budget, unmaintainable due to poor design and documentation, and unreliable due to poor system analysis and testing. Increase the use of object-oriented programming through languages such as C++ and Objective-C.
- Java is developed and released in the mid-1990s. Increasing attention paid to notions of software architecture. Client-server distributed architectures are increasingly used. The UML is proposed.
- Today, the use of integrated development environments becomes more common. Use of stand-alone CASE tools declines. Use of the UML becomes widespread. Increasing use of scripting languages such as Python and PERL for software development. C# developed as a competitor to Java.

1.3 Types of Software

Software is the collection of computer programs, procedures and documentation that performs different tasks on a computer system. The term 'software' was first used by John Tukey in 1958. At the very basic level, computer software consists of a machine language that comprises groups of binary values, which specify processor instructions. The processor instructions change the state of computer hardware in a predefined sequence. Briefly, computer software is the language in which a computer speaks. Computer software can be put into categories based on common function, type, or field of use. There are three broad classifications:

- **Application Software:** It enables the end users to accomplish certain specific tasks. They include programs such as web browsers, office software, games, and programming tools.

- **System Software:** It helps in running computer hardware and the computer system. System software refers to the operating systems; device drivers, servers, windowing systems and utilities (e.g. anti-virus software, firewalls, disk defragmenters).

- **Utility Software:** Small software that usually performs some useful tasks is known as utility software. Example: Win Zip, JPEG Compressor, PDF Merger, PDF to Word Converter etc.

1.4 Software Engineering

- The term 'Software' is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called software product.
- The other term 'Engineering' is all about developing some quality products by using some well-defined, scientific principles and standard within the specified budget and time.
- Hence, Software engineering is defined as: It is a branch of computer science which is associated with development of quality software product using well-defined scientific principles, methods and procedures.
- The outcome of software engineering is an efficient and reliable software product that support over the long term.

1.5 Software process and software process model

- The prime objective of software engineering develop methods for large systems, which produce quality software at low cost and in reasonable time.
- So it is essential to perform software development in phases. This phased development of software is often referred to as software development life cycle (SDLC) or software process and the models used to achieve these goals are termed as Software Process Models.

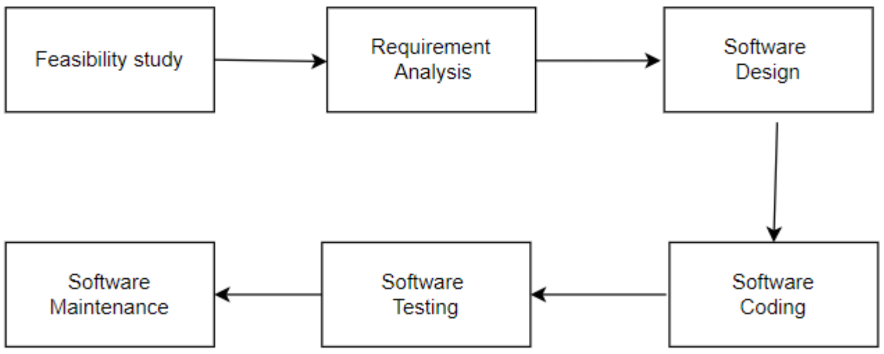


Fig: Software Development Process

1. **Feasibility study/ Preliminary investigation:**
   - Feasibility study decides whether the new system should be developed or not.
   - There are three constraints, which decides the go or no-go decision.

a. Technical:
- determines whether technology needed for proposed system is available or not.
- determines whether the existing system can be upgraded to use new technology whether the organization has the expertise to use it or not.

b. Time:
- determines the time needed to complete a project.
- Time is an important issue as cost increases with an increase in the time period of a project.

c. Budget:
- This evaluation looks at the financial aspect of the project.
- determines whether the investment needed to implement the system will be recovered at later stages or not.

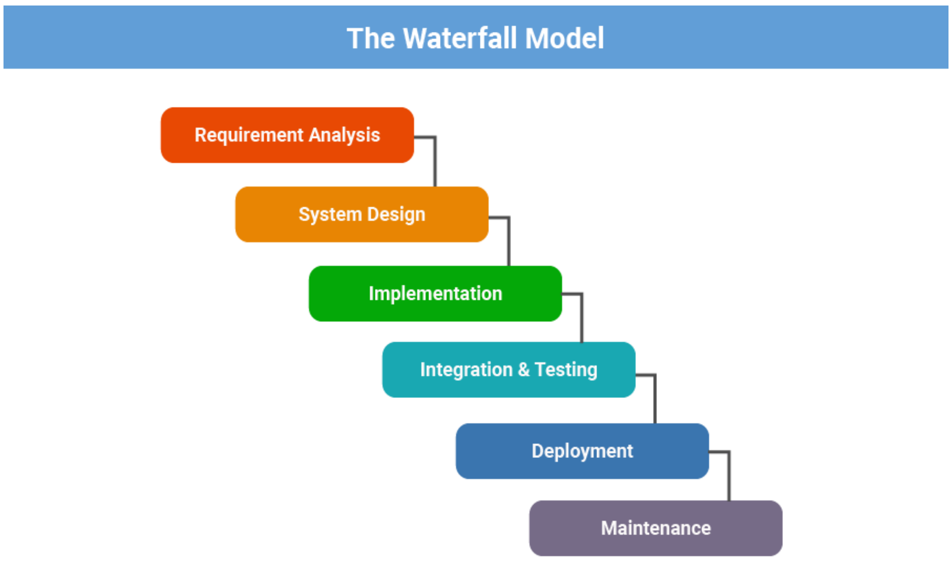
2. **Requirement Analysis/Software Analysis:**
   - Studies the problem or requirements of software in detail.
   - After analyzing and elicitations of the requirements of user, a requirement statement known as **software requirement specification** (SRS) is developed.
3. **Software Design:**
   - Most crucial phase in the development of a system. The SDLC process continues to move from what questions of the analysis phase to the how.
   - Logical design is turned into a physical design.
   - Based on the user requirements and detailed analysis the system must be designed.
   - Input, output, databases, forms, processing specifications etc. are drawn up in detail.
   - Tools and techniques used for describing the system design are: Flowchart, ERD, Data flow diagram (DFD), UML diagrams like Use case, Activity, Sequence etc.
4. **Software Coding:**
   - **Physical design into software code.**
   - Writing a software code requires a prior knowledge of programming language and its tools. Therefore, it is important to choose the appropriate programming language according to the user requirements.
   - A program code is efficient if it makes optimal use of resources and contains minimum errors.
5. **Software Testing:**
   - **Software testing is performed to ensure that software produces the correct outputs i.e. free from errors.** This implies that outputs produced should be according to user requirements.
   - Efficient testing improves the quality of software.
   - Test plan is created to test software in a planned and systematic manner.
6. **Software Maintenance:**
   - This phase comprises of a set of software engineering activities that occur after software is delivered to the user.
   - After the software is developed and delivered, it may require changes. Sometimes, changes are made in software system when user requirements are not completely met.
   - To make changes in software system, software maintenance process evaluates, controls, and implements changes.

Software process:

- When you work to build a product or system, it’s important to go through a series of predictable steps—a road map that helps you to create a timely, high-quality result. The road map that you follow is called a “software process.”
- Software engineers and their managers adapt the process to their needs and then follow it. In addition, the people who have requested the software have in the process of defining, building, and testing it.
- Tt provides path, stability, control over your project.

a. The Waterfall Model

The waterfall model, sometimes called the classic life cycle, suggests a systematic, Sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, culminating in ongoing support of the completed software.



The linear process flow model will be appropriate when:

- User requirements are clearly defined
- Less number of staffing to do simultaneous work on that project in the organization
- No much re-useable components
- Sufficient time for software development
- Less communication between customer and developer
- Software engineer have clear idea to do the prescribed task

The waterfall model has the following six stages:

- **Requirement Analysis:** The model's initial stage involves defining all the project's function and nonfunctional requirements. Functional requirements pertain to the project's external fulfillments, such as obtaining credit-card validation for activating a free trial. On the other hand, nonfunctional requirements involve technicalities, such as measuring performance.
- **System Design:** After the requirements have been analyzed, they can be transformed into a design or flow chart. The project specifications are created and studied to determine what the final project will look like, along with the required actions to get there.
- **Implementation:** In software development, this stage is where the actual coding begins. All the work compiled in the previous steps is applied to this stage to start implementing the project findings.
- **Integration and Testing:** After the coding is completed, testing rounds begin. The product is not prepared to be delivered to the customer until all bugs have been sorted out. An optional user acceptance test (UAT) might be performed at this stage, which involves a small group of initial users testing the product.
- **Deployment:** The deployment stage involves the release of the software. The product is complete, and the development team releases the deliverables.
- **Maintenance:** After the product has been released to the public, new issues may be brought to the developer's attention. The development team remains responsible for providing solutions to the problems and updating the software based on the feedback.

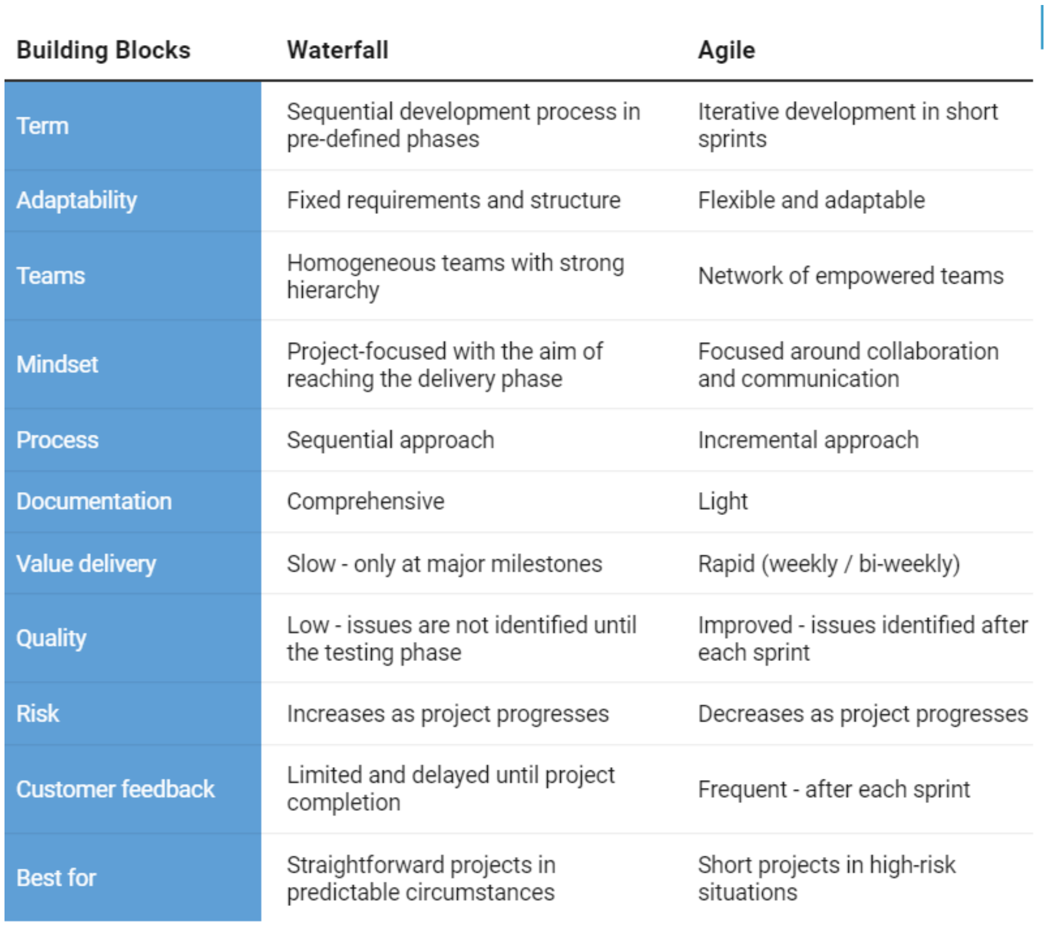
All these stages flow downwards one-by-one like a waterfall. The next stage is started only after the previous one is over.

Advantages of Waterfall Model

- Before the next project phase starts, each previous phase must be completed, meaning early roadblocks elimination and more polished products in the end
- Determines the end goal early, meaning strong commitment to the end result
- Before each stage is completed, quality assurance tests must be taken, meaning fewer chances of issues and roadblocks
- Elaborate documentation is compiled at every stage to ensure quality results
- Project is completely dependent on the project team with minimum client intervention, meaning teams work their way through to the project goal without frequent fixes and adjustments from the client's side.

Disadvantages of Waterfall Model

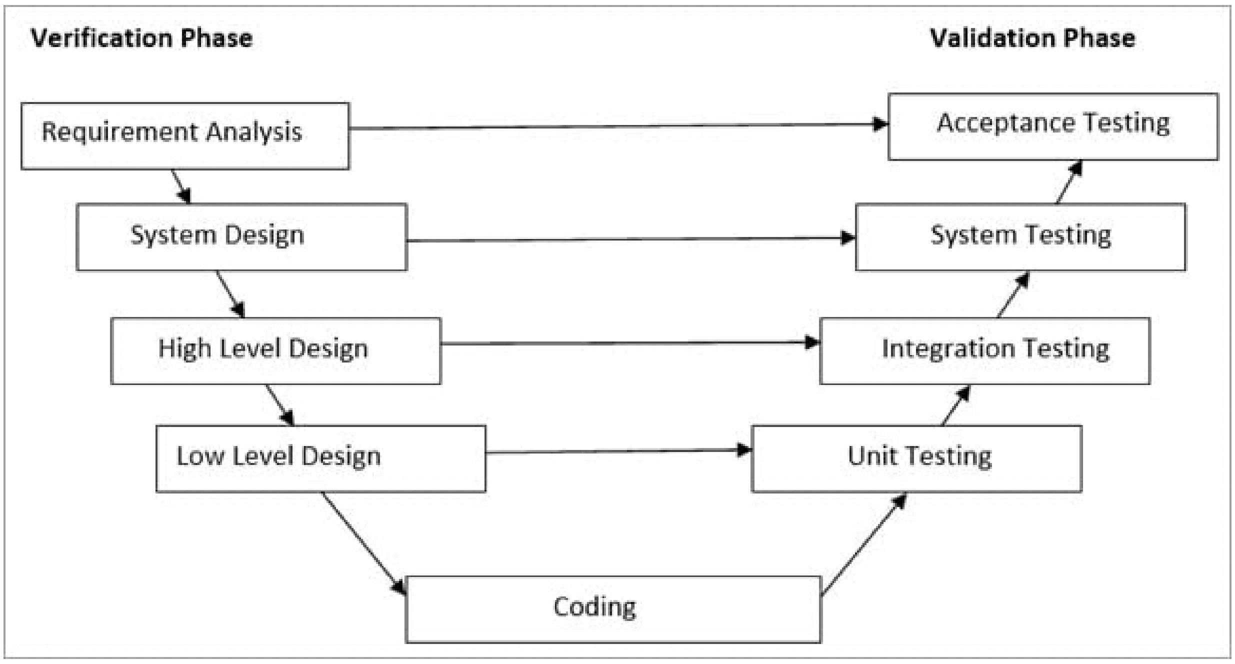
- You can't apply this model to complex projects with changing requirements due to the rigid sequential approach
- This approach excludes the client from the development process, meaning high chances of late change requests and unmet deadlines
- The testing period comes late in the development process and doesn't allow simultaneous development and testing to fix issues at the early stages
- Documentation writing takes a lot of time
- You can't use feedback from your clients to adjust the product to their needs, meaning lower customer satisfaction



Building Blocks	Waterfall	Agile
Term	Sequential development process in pre-defined phases	Iterative development in short sprints
Adaptability	Fixed requirements and structure	Flexible and adaptable
Teams	Homogeneous teams with strong hierarchy	Network of empowered teams
Mindset	Project-focused with the aim of reaching the delivery phase	Focused around collaboration and communication
Process	Sequential approach	Incremental approach
Documentation	Comprehensive	Light
Value delivery	Slow - only at major milestones	Rapid (weekly / bi-weekly)
Quality	Low - issues are not identified until the testing phase	Improved - issues identified after each sprint
Risk	Increases as project progresses	Decreases as project progresses
Customer feedback	Limited and delayed until project completion	Frequent - after each sprint
Best for	Straightforward projects in predictable circumstances	Short projects in high-risk situations

b. V-model

V- Model is also known as Verification and Validation Model. In this model Verification & Validation goes hand in hand i.e. development and testing goes parallel. V model and waterfall model are the same except that the test planning and testing start at an early stage in V-Model.



<u>**V-Model Verification Phases**</u>

i. **Requirement Analysis:** In this phase, all the required information is gathered & analyzed. Verification activities include reviewing the requirements.
ii. **System Design:** Once the requirement is clear, a system is designed i.e. architecture, components of the product are created and documented in a design document.
iii. **High-Level Design:** High-level design defines the architecture/design of modules. It defines the functionality between the two modules.
iv. **Low-Level Design:** Low-level Design defines the architecture/design of individual components.
v. **Coding:** Code development is done in this phase.

<u>**V-Model - Validation Phases**</u>

i. **Unit Testing:** Unit testing is performed using the unit test cases that are designed and is done in the Low-level design phase. Unit testing is performed by the developer itself. It is performed on individual components which lead to early defect detection.
ii. **Integration Testing:** Integration testing is performed using integration test cases in High-level Design phase. Integration testing is the testing that is done on integrated modules. It is performed by testers.
iii. **System Testing:** System testing is performed in the System Design phase. In this phase, the complete system is tested i.e. the entire system functionality is tested.
iv. **Acceptance Testing:** Acceptance testing is associated with the Requirement Analysis phase and is done in the customer's environment.

<u>**When to use V-Model?**</u>

- When the requirement is well defined and not ambiguous.
- The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
- The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.

<u>Advantages of V - Model</u>

- It is a simple and easily understandable model.
- Testing Methods like planning, test designing happens well before coding.
- Works well for small plans where requirements are easily understood.
- V - Model approach is good for smaller projects wherein the requirement is defined and it freezes in the early stage.
- It is a systematic and disciplined model which results in a high-quality product.

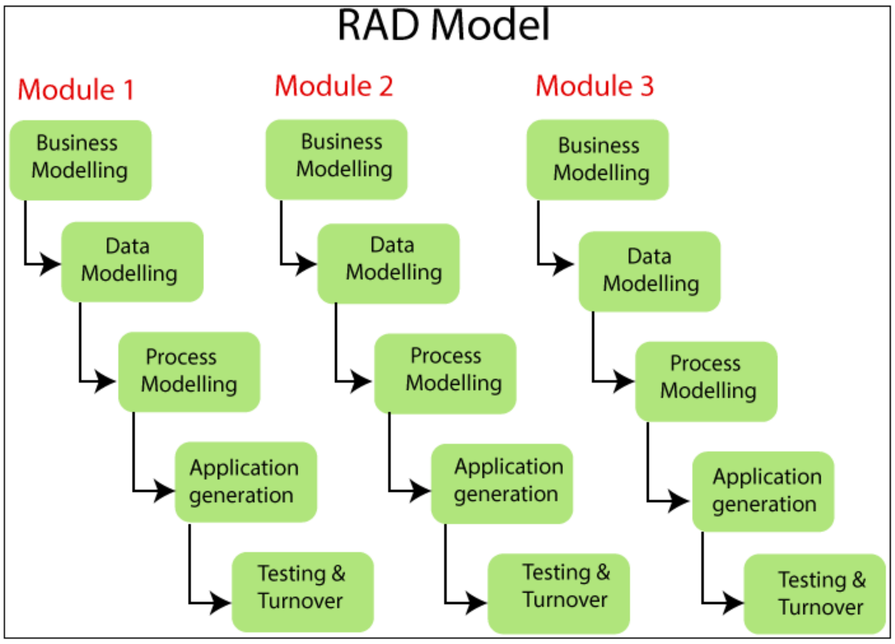
<u>Disadvantages of V-Model</u>

- Very rigid and least flexible.
- Not a good for a complex project.
- V-shaped model is not good for ongoing projects.
- Requirement change at the later stage would cost too high.

c. Rapid Application Development (RAD) Model

It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype.

This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.



The RAD model will be appropriate when:

i. When budget is high enough to afford the development cost.
ii. We have to develop the application in short period of time.
iii. Customer clearly defined their requirements.
iv. No much re-useable components or no chances of outsourcing in organization.

<u>Phases of RAD Model</u>

Business modeling: The information flow is identified between various business functions.

Data modeling: Information gathered from business modeling is used to define data objects that are needed for the business.

Process modeling: Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.

Application generation: Automated tools are used to convert process models into code and the actual system.
Testing and turnover: Test new components and all the interfaces.

Advantages of RAD Model:

- Increases re-usability of components.
- Less development time.
- Quick initial responses.
- Encourages customer feedbacks.
- Involvement of end users from very beginning solves lot of development issues.

Disadvantages of RAD Model:

- It requires user participation throughout the life cycle.
- Requires strong and skilled developer's team.
- Cost is very high.

c. Evolutionary Process Models

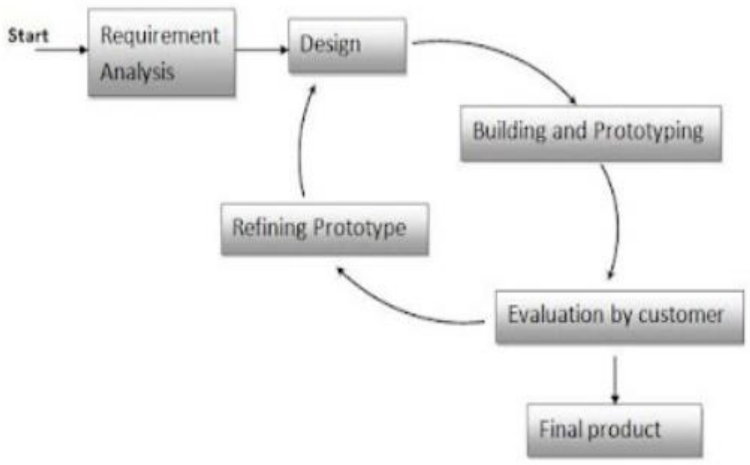
Evolutionary models are iterative which is much suitable for new systems where no clear Idea of the requirements, inputs and outputs parameters. It produces an increasingly more complete version of the software with each iteration. It is combination of Iterative and Incremental model of software development life cycle.
The evolutionary process models are appropriate in following situation:

- No clear idea about the product for both customers and developers.
- Good communication between the customers and developers.
- Sufficient time for the software development
- Less chances of outsourcing and availability of re-usable components.

The two common evolutionary process models are:

i. Prototyping Model

Prototyping Model is a software development model in which prototype is built, tested, and reworked until an acceptable prototype is achieved. It also creates base to produce the final system or software. It works best in scenarios where the project's requirements are not known in detail. It is an iterative, trial and error method which takes place between developer and client.



<u>Phases of Prototyping Models</u>

a. **Requirements gathering and analysis:** A prototyping model starts with requirement analysis. In this phase, the requirements of the system are defined in detail. During the process, the users of the system are interviewed to know what their expectation from the system is.
b. **Quick design:** The second phase is a preliminary design or a quick design. In this stage, a simple design of the system is created. However, it is not a complete design. It gives a brief idea of the system to the user. The quick design helps in developing the prototype.
c. **Build a Prototype:** In this phase, an actual prototype is designed based on the information gathered from quick design. It is a small working model of the required system.
d. **Initial user evaluation:** In this stage, the proposed system is presented to the client for an initial evaluation. It helps to find out the strength and weakness of the working model. Comment and suggestion are collected from the customer and provided to the developer.
e. **Refining prototype:** If the user is not happy with the current prototype, you need to refine the prototype according to the user's feedback and suggestions. This phase will not over until all the requirements specified by the user are met. Once the user is satisfied with the developed prototype, a final system is developed based on the approved final prototype.
f. **Implement Product and Maintain:** Once the final system is developed based on the final prototype, it is thoroughly tested and deployed to production. The system undergoes routine maintenance for minimizing downtime and prevent large-scale failures.

<u>Advantages of Prototyping Model</u>

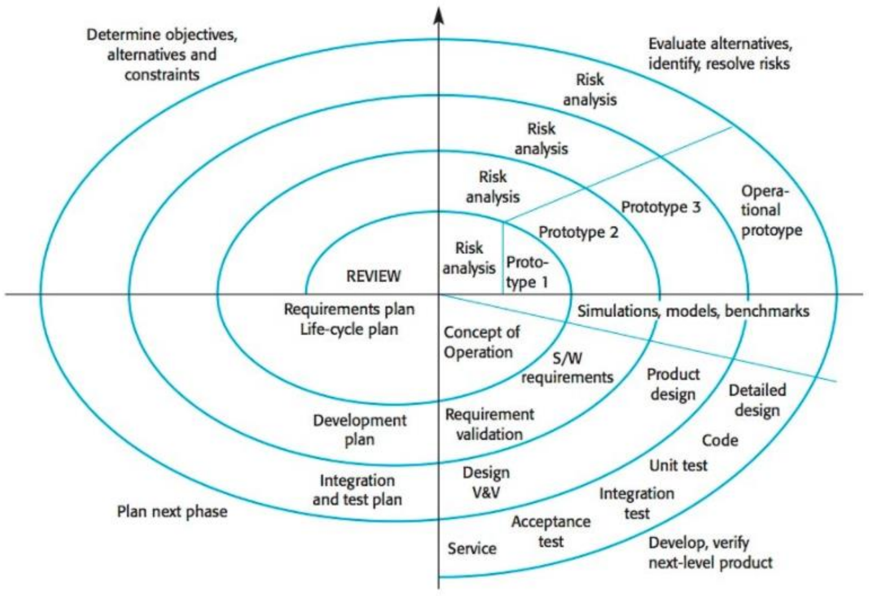
- Users are actively involved in the development.
- Errors can be detected much earlier.
- Reduce Maintenance cost.
- Confusing or difficult functions can be identified
- Quicker user feedback is available leading to better solutions.
- It identifies the missing functionality easily. It also identifies the confusing or difficult functions.

<u>Disadvantages of Prototyping Model</u>

- The client involvement is more and it is not always considered by the developer.
- It is a slow process because it takes more time for development.
- Difficult to know how long the project will last.
- Many changes can disturb the rhythm of the development team.
- Poor documentation because the requirements of the customers are changing.

ii. The Spiral Model.

The spiral model was originally proposed by Barry Boehm, is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model. It provides the potential for rapid development of increasingly more complete versions of the software.



Spiral Model Phases

- **Planning:** The objectives, alternatives and constraints of the project are determined and are documented.
- **Risk Analysis:** All possible alternatives which can help in developing a cost effective project are analyzed. This phase identify and resolve all the possible risks in the project management.
- **Engineering (Development and validation):** It includes testing, coding and deploying software at the customer site. The actual development of the project is carried out. The output of this phase is passed through all the phases iteratively in order to obtain improvements in the same.
- **Evaluation:** Review the results achieved so far with the customer and plan the next iteration around the spiral. Progressively more complete version of the software gets built with each iteration around the spiral.

When to use Spiral Model?

- It is used when project is large
- When creation of a prototype is applicable
- When risk and costs evaluation is important
- Useful for medium to high-risk projects
- When requirements are unclear and complex
- When changes may require at any time
- When long term project commitment is not feasible due to changes in economic priorities

Advantages of Spiral Model

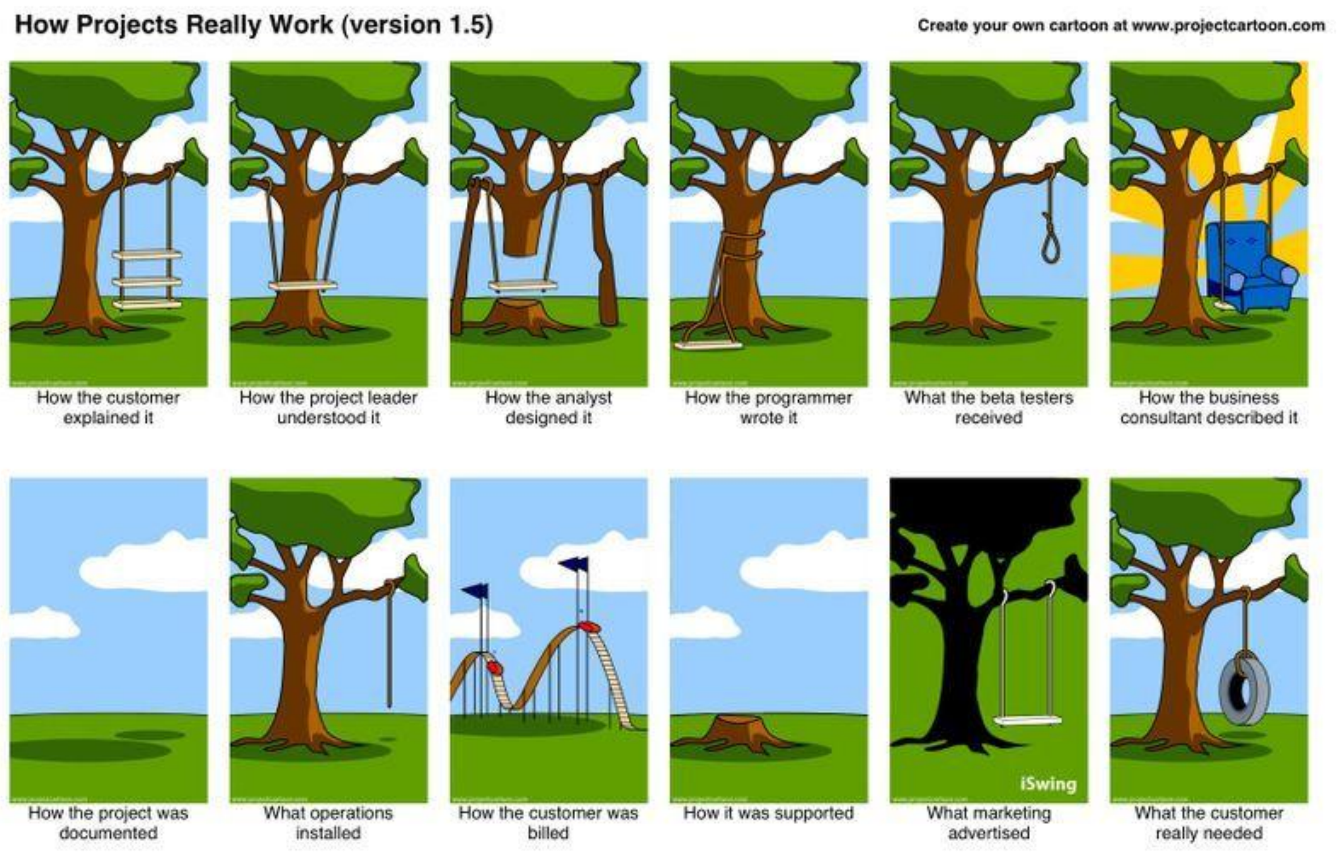
- Additional functionality or changes can be done.
- Cost estimation becomes easy as the prototype buildings is done in fragments.
- Continuous or repeated development helps in risk management.
- Early and frequent feedback from users.
- Users see the system early.

Disadvantages of Spiral Model

- End of project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex.
- Documentation is more as it has more intermediate phases.
- Risk assessment expertise is required.

The Problems with our Requirements Practices

• We have trouble understanding the requirements that we do acquire from the customer
• We often record requirements in a disorganized manner
• We spend far too little time verifying what we do record
• We allow change to control us, rather than establishing mechanisms to control change
• Most importantly, we fail to establish a solid foundation for the system or software that the user wants built


How Projects Really Work (version 1.5)
Create your own cartoon at www.projectcartoon.com
How the customer explained it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
What the beta testers received
How the business consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
iSwing
What marketing advertised
What the customer really needed

Solution: Requirements Engineering

• Begins during the communication activity and continues into the modeling activity
• Builds a bridge from the system requirements into software design and construction
• Allows the requirements engineer to examine
– The context of the software work to be performed
– The specific needs that design and construction must address
– The priorities that guide the order in which work is to be completed
– The information, function, and behavior that will have a profound impact on the resultant design

Requirement Engineering

– It is the process of defining, documenting, and maintaining requirements in the engineering design process.
– It helps software engineer to better understand the problem they will work to solve.
– Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system.
– Participant: Software Engineers, managers, customers and end users
– It is a software engineering action that begin during the communication activity and continues into the modeling activity
– Requirement Engineering provides the appropriate mechanism for

- Understanding what the customer want
- Analyzing need
- Assessing feasibility
- Negotiating a reasonable solution
- Specifying a solution unambiguously
- Validating the specification
- Managing the requirement as they are transformed into an operational system

Requirements engineering process:
– Requirements engineering process of the understanding & defining what services are required from the system and identifying the constraints on the system development.
–The goal of requirement engineering process is to create and maintain a system requirements documents.

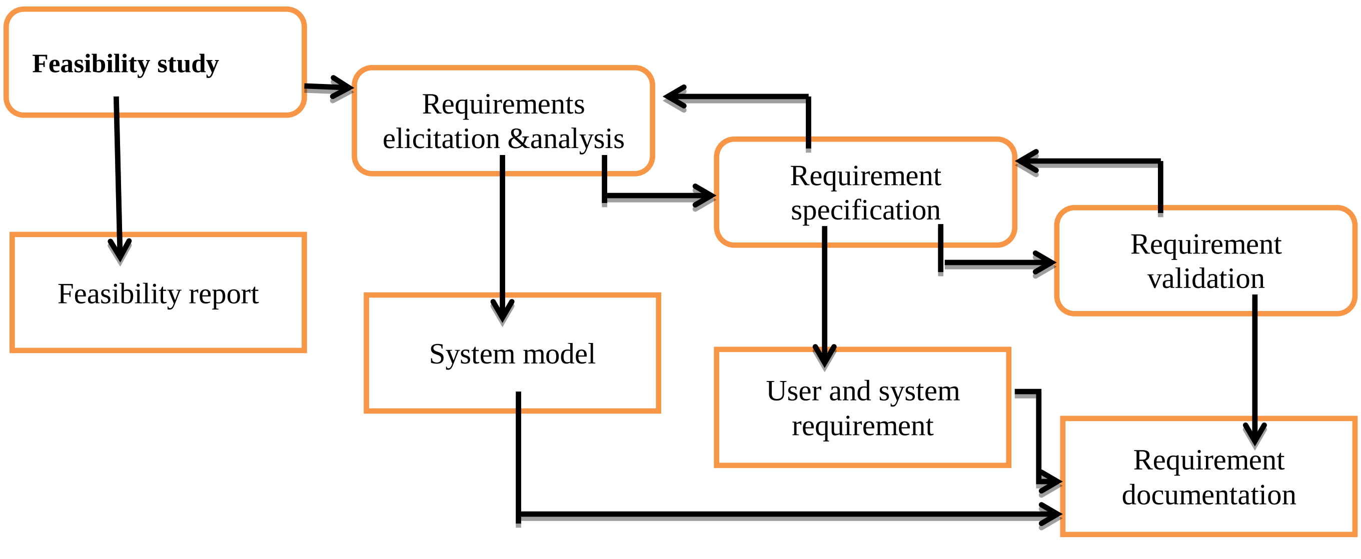

Feasibility study
Requirements elicitation &analysis
Requirement specification
Requirement validation
Feasibility report
System model
User and system requirement
Requirement documentation

Fig: requirement engineering process

Main activities in engineering requirement process:

1. Feasibility study:

- A feasibility study decides whether or not the proposed system is worthwhile.
    - A short focused study that checks:
        - If the system contributes to organizational objectives;
        - If the system can be implemented using current technology, within given cost and schedule constraints;
        - If the system can be integrated with other systems that are already in place.
- An estimate is made whether the user need may be satisfied using current software and hardware technologies.
- The study also considers whether the purposed system is cost-effective from business point of view.
- It should be quick and cheap.
- Should provide information to decide whether or not to go ahead with more detail analysis.

Feasibility study implementation

• Feasibility study involves information assessment (what is required), information collection and report writing.

• Questions for people in the organisation for information assessment and collection:

– What if the system wasn't implemented?
– What are current process problems?
– How will the proposed system help?
– What will be the integration problems?
– Is new technology needed? What skills?
– What facilities must be supported by the proposed system?

• Feasibility study report should make a recommendation about the development to continue or not.

Types of Feasibility:

a. **Technical Feasibility** - Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.
b. **Operational Feasibility** - Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.
c. **Economic Feasibility** - Economic feasibility decides whether the necessary software can generate financial profits for an organization.

2. Requirement elicitation and analysis:

- "**What do the system stakeholders (end-users) require or expect from the system**?"
- Derivation of the system requirements by observing the existing system, discussion with potential users & procurers, task analysis.
- Involve development of one or more system models & prototype.
- Helps us to understand the system to be specified.
- It deals with all the activities required in gathering the requirements of a system.
- The developers and engineers work in close coordination with the customers and end-users to identify more about the problem to be solved and to bridge the gap between expectation and understanding.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called stakeholders.
- Eliciting & understanding stakeholder requirement is difficult due to the following reasons:
    - Stakeholders don't know what they really want except in most general.
    - Stakeholders express requirements in their own terms.
    - Different stakeholders may have deferent requirements.
    - Organizational and political factors may influence the system requirements.
    - The requirements change during the analysis process. New stakeholders may emerge and the business environment change.

3. Requirement specification:

- Activity of translating the information gathered during the analysis activity into document that defines the set of requirement.
- Process of writing down the user and system requirements in a required document.
- Ideally the user and system requirements should be clear, unambiguous, easy to understand, complete and consistent. Practically, it is almost impossible to achieve.

Two types of requirement are:

i) user requirement :
   - Abstract statements of the system requirements for the customer and end-user of the system.
   - It should describe functional and nonfunctional requirements so that they are understandable by system users who don't have detailed technical knowledge.

ii) system requirement :
   - more detailed description of the functionality to be provided
   - Expanded version of the users requirements that software engineers use as the starting point for the system design.

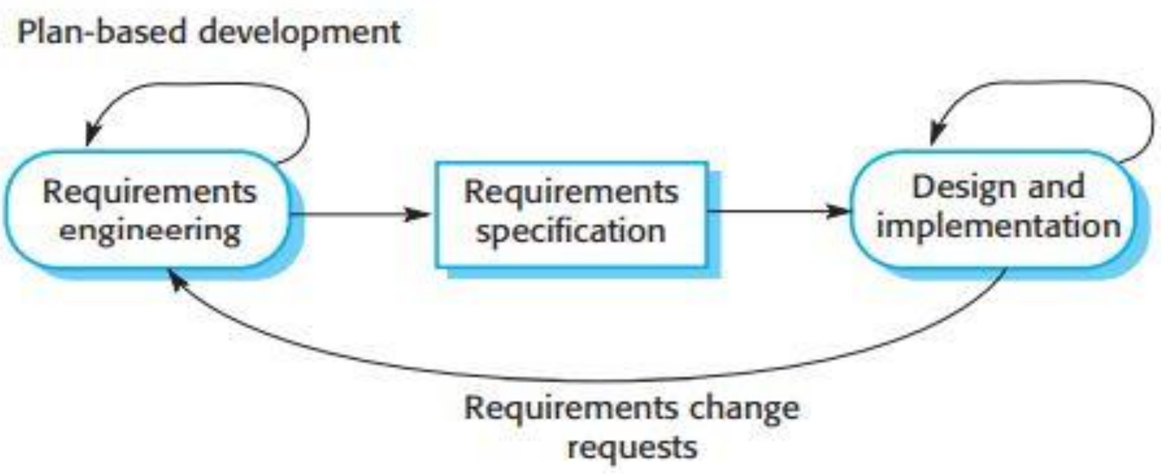
4. Requirement validation:

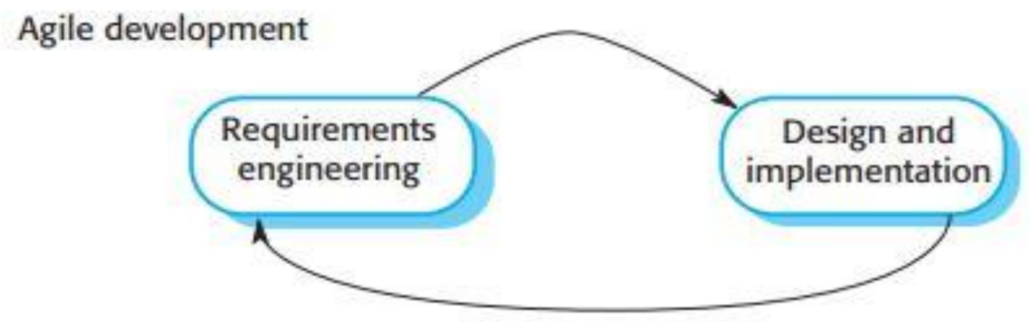
- The process of checking the requirements define the system that the customer really wants.
- Checks the realism, consistency, completeness of requirements.
- Errors in the requirements are discovered & modified to correct these problems.
- It overlaps with elicitation and analysis as it is concerned with finding problems with the requirements.
- It is critically important because errors in a requirement document can lead to extensive rework costs when these problems are discovered during development or after system is in service.

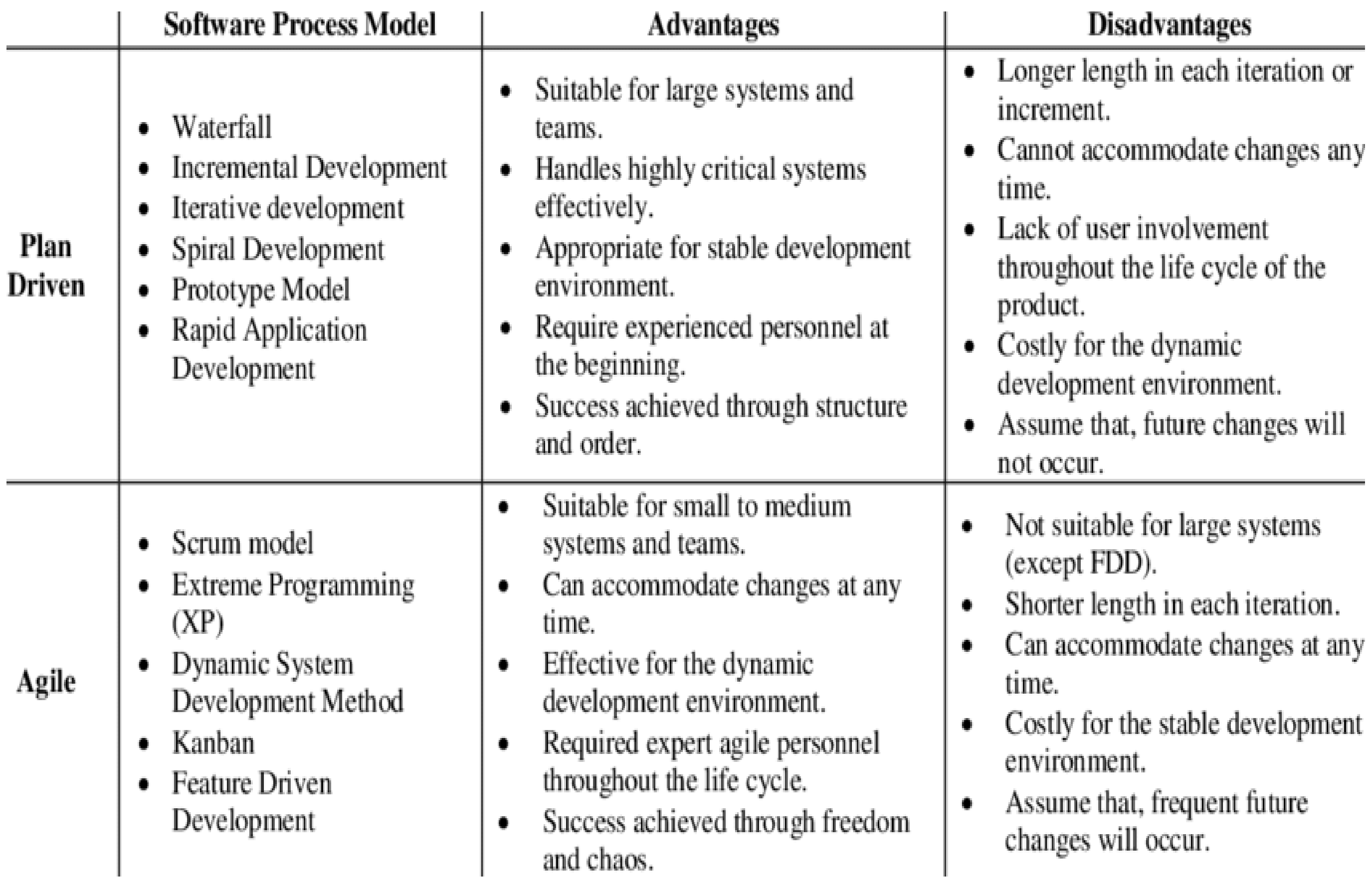
Agile Software Development

- Rapid development and delivery is now often the most important requirement for software systems. Businesses operate in a fast-changing requirement and it is practically impossible to produce a set of stable software requirements. Software has to evolve quickly to reflect changing business needs.
- Agile development methods emerged in the late 1990s whose aim was to radically reduce the delivery time for working software systems.
- It is a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. In the Agile model in software testing, both development and testing activities are concurrent, unlike the Waterfall model.
- It 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.

Plan Based VS Agile Software Development


Plan-based development
Requirements engineering
Requirements specification
Design and implementation
Requirements change requests





	Software Process Model	Advantages	Disadvantages
Plan Driven	• Waterfall • Incremental Development • Iterative development • Spiral Development • Prototype Model • Rapid Application Development	• Suitable for large systems and teams. • Handles highly critical systems effectively. • Appropriate for stable development environment. • Require experienced personnel at the beginning. • Success achieved through structure and order.	• Longer length in each iteration or increment. • Cannot accommodate changes any time. • Lack of user involvement throughout the life cycle of the product. • Costly for the dynamic development environment. • Assume that, future changes will not occur.
Agile	• Scrum model • Extreme Programming (XP) • Dynamic System Development Method • Kanban • Feature Driven Development	• Suitable for small to medium systems and teams. • Can accommodate changes at any time. • Effective for the dynamic development environment. • Required expert agile personnel throughout the life cycle. • Success achieved through freedom and chaos.	• Not suitable for large systems (except FDD). • Shorter length in each iteration. • Can accommodate changes at any time. • Costly for the stable development environment. • Assume that, frequent future changes will occur.

Agile methods

- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods. (Plan-driven approaches involve a significant overhead in planning, designing, and documenting the system.). These methods:
    - Focus on the code/software itself rather than the design and documentation.
    - Are based on an iterative approach to software development.
    - Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.
- The aim of agile methods is to reduce overheads in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework.

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Principles of Agile Methods
- **Customer involvement:** Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
- **Incremental delivery:** The software is developed in increments with the customer specifying the requirements to be included in each increment.
- **People not process:** The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
- **Embrace change:** Expect the system requirements to change and so design the system to accommodate these changes.
- **Maintain simplicity:** Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

Agile method applicability:

- Product development where a software company is developing a small or medium-sized product.
- Custom system development within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are not a lot of external rules and regulations that affect the software.
- Because of their focus on small, tightly-integrated teams, there are problems in scaling agile methods to large systems.

Problems with Agile Methods
- It can be difficult to keep the interest of customers who are involved in the process.
- Team members may be unsuited to the intense involvement that characterizes agile methods.
- Prioritizing changes can be difficult where there are multiple stakeholders.
- Maintaining simplicity requires extra work.

- Contracts may be a problem as with other approaches to iterative development.

When to use the Agile Model?

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

Advantages of Agile Method

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

Disadvantages of Agile Model

- 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.
- 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.

Agile Process Model

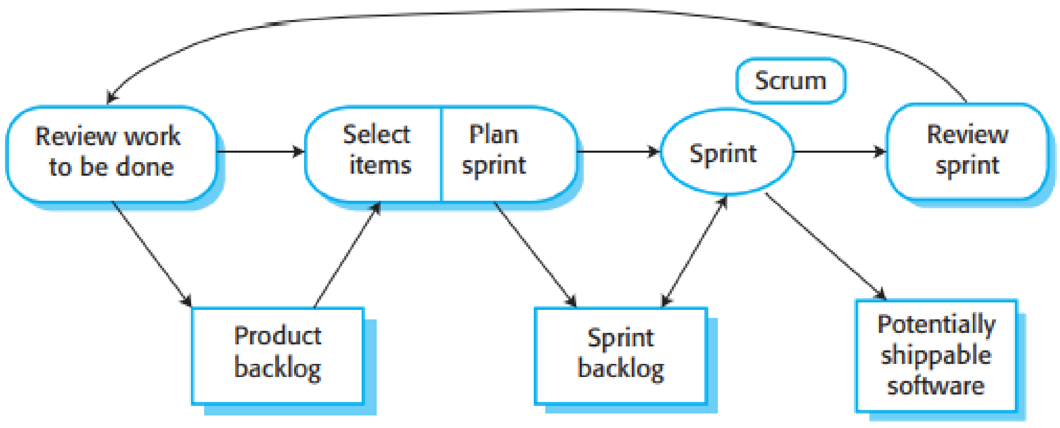
i. Scrum
ii. Crystal
iii. Dynamic Software Development Method(DSDM)
iv. Feature Driven Development(FDD)
v. Lean Software Development
vi. eXtreme Programming(XP)

Scrum:

- It is an agile development method which concentrates specifically on how to manage tasks within a team-based development environment.
- It is a combination of both Incremental and Iterative model for managing product development.
- There are three roles in it, and their responsibilities are:
    - **Scrum Master:** The scrum can set up the master team, arrange the meeting and remove obstacles for the process
    - **Product owner:** The Product Owner usually represents the Client and acts as a point of contact from the Client side. The one who prioritizes the list of Product Backlogs that Scrum Team should finish and release.
    - **Scrum Team:** The team manages its work and organizes the work to complete the sprint or cycle. A cross-functional, self-organizing group of dedicated people (Group of Product Owner, Business Analyst, Developer's and QA's). The recommended size of a scrum team is 7 plus or minus 2 (i.e., between 5 to 9 members in a team).

Process flow of Scrum

- Each iteration of a scrum is known as Sprint
- Product backlog is a list where all details are entered to get the end-product
- During each Sprint, top user stories of Product backlog are selected and turned into Sprint backlog
- Team works on the defined sprint backlog
- Group of user stories which scrum development team agreed to do during the current sprint (Committed Product Backlog items)
- Team checks for the daily work
- At the end of the sprint, team delivers product functionality



In an Agile Scrum Methodology, all the members in a Scrum Team gathers and finalize the Product Backlog Items (User Stories) for a particular Sprint and commit the timeline to release the product. Based on the Daily Scrum meetings, Scrum Development Team develops and tests the product and presents it to the Product Owner at Sprint Review Meeting. If the Product Owner accepts all the developed User Stories then the Sprint is completed and the Scrum Team goes for the next Sprint in a same manner.

When do we use Agile Scrum Methodology?

- The client is not so clear on the requirements.
- The client expects quick releases.
- The client doesn’t give all the requirements at a time.

System Modeling:

- System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.
- It is about representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML).
- It helps the analyst to understand the functionality of the system and models are used to communicate with customers.

Models can explain the system from different perspectives:

- An **external perspective**, where you model the context or environment of the system.[**Context Models]**
- An **interaction perspective**, where you model the interactions between a system and its environment, or between the components of a system. **[Interaction Models]**
- A **structural perspective**, where you model the organization of a system or the structure of the data that is processed by the system. **[Structural Models]**
- A **behavioral perspective**, where you model the dynamic behavior of the system and how it responds to events. **[Behavioral Model]**

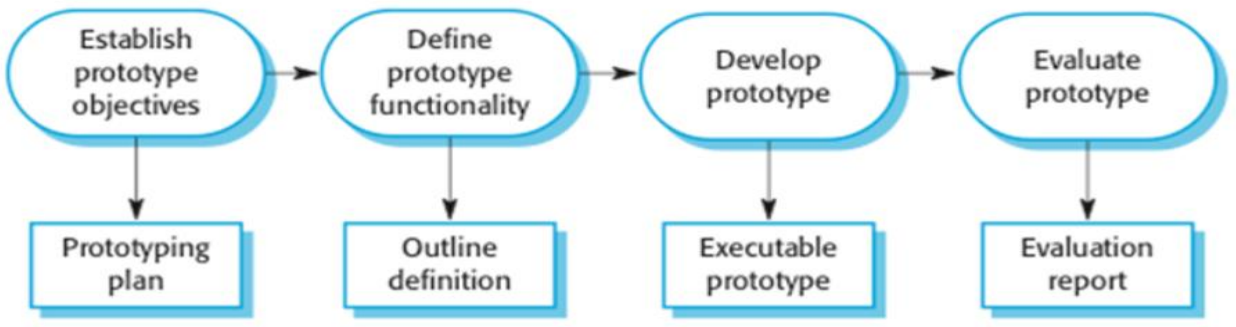
Types of UML diagrams for system modeling

- **Activity diagrams:** It shows the activities involved in a process or in data processing.
- **Use case diagrams:** It shows the interactions between a system and its environment.
- **Sequence diagrams:** It shows interactions between actors and the system and between system components.
- **Class diagrams:** It shows the object classes in the system and the associations between these classes.
- **State diagrams:** It shows how the system reacts to internal and external events.

Software Prototyping:

- Prototyping is the rapid development of a system. In the past, the prototype was normally thought of as inferior in some way to the required system, so further development was required.
- Now, the boundary between prototyping and normal system development is blurred and many systems are developed using an evolutionary approach.
- The principal use is to help customers and developers understand the requirements for the system.
  - **Requirements elicitation:** Users can experiment with a prototype to see how the system supports their work
  - **Requirements validation:** The prototype can reveal errors and omissions in the requirements.
- Prototyping can be a risk reduction activity that reduces requirements risks.

Phases of software Prototyping



The phases of software prototyping are:

1. **Establish objectives:** The objective of the prototype should be made clear from the start of the process. Is it to validate system requirements or demonstrate feasibility, etc.
2. **Define prototype functionality:** Decide what are the inputs and the expected output from a prototype. To reduce the prototyping costs and accelerate the delivery schedule, you may ignore some functionality, such as response time and memory utilization unless they are relevant to the objective of the prototype.
3. **Develop prototype:** The initial prototype is developed that includes only user interfaces.
4. **Evaluate prototype:** Once the users are trained to use the prototype, they then discover requirements errors. Using the feedback both the specifications and the prototype can be improved. If changes are introduced, then a repeat of steps 3 and 4 may be needed

Prototyping benefits

- Misunderstandings between software users and developers are exposed.
- Missing services may be detected and confusing services may be identified.
- A working system is available early in the process.
- The prototype may serve as a basis for deriving a system specification.
- The system can support user training and system testing.
- Improved system usability
- Closer match to the system needed
- Improved design quality
- Improved maintainability
- Reduced overall development effort

Prototyping objectives

- To demonstrate that the prototype has been developed according to the specification and that the final specification is appropriate.
- To collect information about errors or other problems in the system such as user interface problems that need to be addressed in the intermediate prototype stage.
- To give management and everyone connected with the project the first glimpse of what the technology can provide.