Module 2

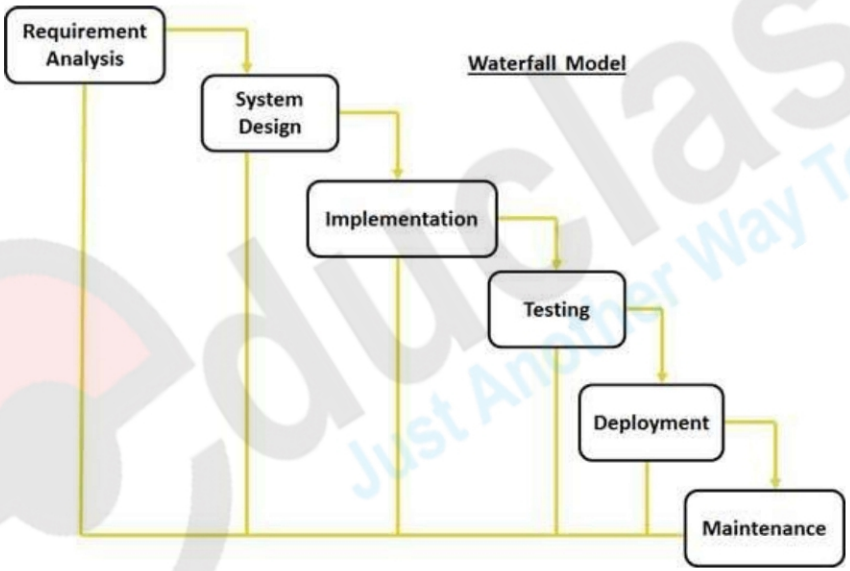
Software Process Models

Waterfall Model:

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.



Waterfall Model

The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Pros of Waterfall Model

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model . each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Cons of Waterfall Model

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

Evolutionary Process Model

Prototype Model

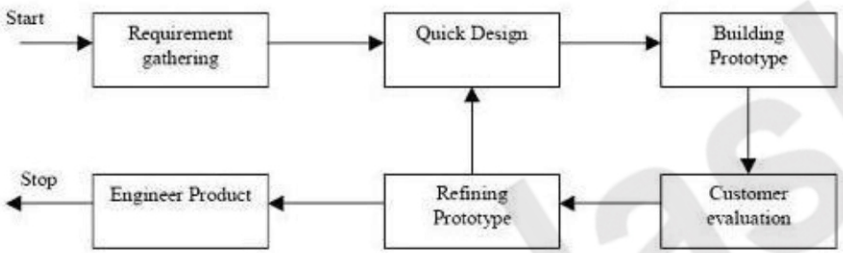
The Software Prototyping refers to building software application prototypes which display the functionality of the product under development but may not actually hold the exact logic of the original software.

Software prototyping is becoming very popular as a software development model, as it enables to understand customer requirements at an early stage of development. It helps get valuable feedback from the customer and helps software designers and developers understand about what exactly is expected from the product under development.

What is Software Prototyping?

- Prototype is a working model of software with some limited functionality.
- The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.
- Prototyping is used to allow the users evaluate developer proposals and try them out before implementation.
- It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.

Following is the stepwise approach to design a software prototype:



Prototyping Model

- **Requirement Gathering & Identification:** This step involves understanding the very basics product requirements especially in terms of user interface. The more intricate details of the internal design and external aspects like performance and security can be ignored at this stage.

- **Developing the initial Prototype:** The initial Prototype is developed in this stage, where the very basic requirements are showcased and user interfaces are provided. These features may not exactly work in the same manner internally in the actual software developed and the workarounds are used to give the same look and feel to the customer in the prototype developed.

- **Customer Evaluation of the Prototype:** The prototype developed is then presented to the customer and the other important stakeholders in the project. The feedback is collected in an organized manner and used for further enhancements in the product under development.

- **Refine and enhance the Prototype:** The feedback and the review comments are discussed during this stage and some negotiations happen with the customer based on factors like, time and budget constraints and technical feasibility of actual implementation. The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until customer expectations are met.

Pros of Prototyping Model

- Increased user involvement in the product even before implementation
- Since a working model of the system is displayed, the users get a better understanding of the system being developed.
- Reduces time and cost as the defects can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified

Cons of Prototyping Model

- Risk of insufficient requirement analysis owing to too much dependency on prototype
- Users may get confused in the prototypes and actual systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Developers may try to reuse the existing prototypes to build the actual system, even when it's not technically feasible
- The effort invested in building prototypes may be too much if not monitored properly

Spiral Model

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model.

Spiral model is a combination of iterative development process model and sequential linear development model i.e. waterfall model with very high emphasis on risk analysis.

It allows for incremental releases of the product, or incremental refinement through each iteration around the spiral.

Spiral Model design

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

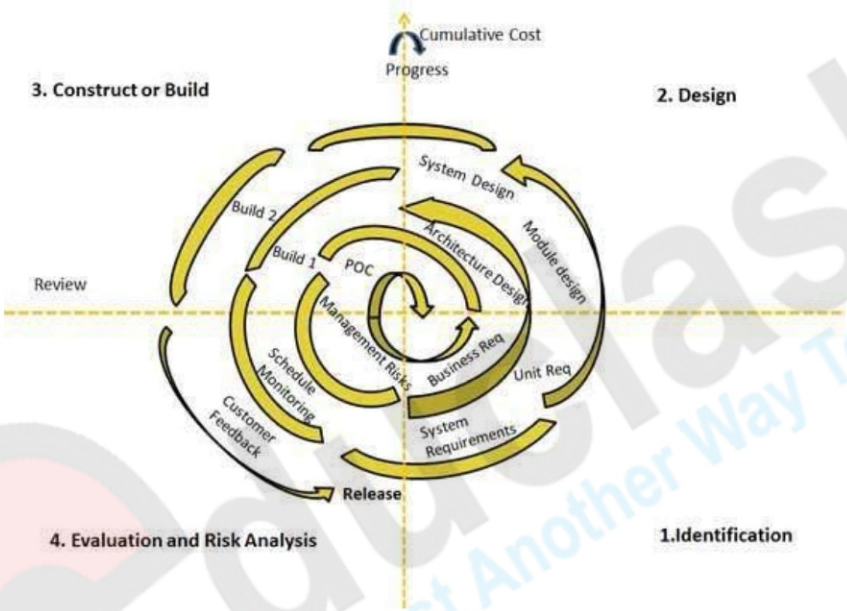
- **Identification:** This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase. This also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral the product is deployed in the identified market.

- **Design:** Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and final design in the subsequent spirals.

- **Construct or Build:** Construct phase refers to production of the actual software product at every spiral. In the baseline spiral when the product is just thought of and

the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback. Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to customer for feedback.

- **Evaluation and Risk Analysis:** Risk Analysis includes identifying, estimating, and monitoring technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.



Based on the customer evaluation, software development process enters into the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.

Pros of Spiral Model

- Changing requirements can be accommodated.
- Allows for extensive use of prototypes
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.

Cons of Spiral Model

- Management is more complex.
- 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
- Spiral may go indefinitely.
- Large number of intermediate stages requires excessive documentation.

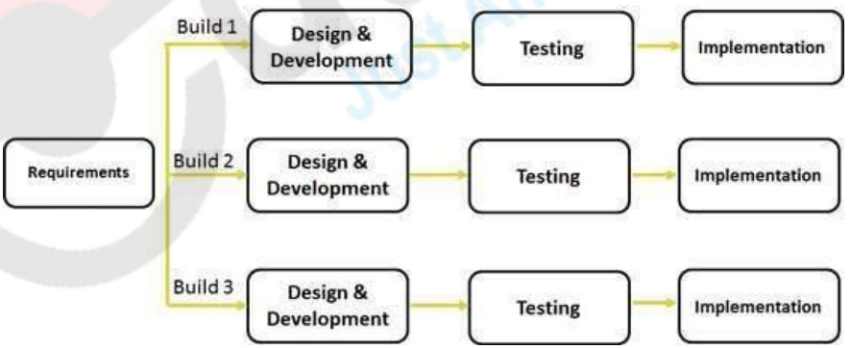
Iterative Approach:

In Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

Iterative Model design

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).



Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." and "This process may be described as an "evolutionary acquisition" or "incremental build" approach."

In incremental model the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation

and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement.

The key to successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model. As the software evolves through successive cycles, tests have to be repeated and extended to verify each version of the software.

Pros of Iterative Model

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment operational product is delivered.
- Issues, challenges & risks identified from each increment can be utilized/applied to the next increment.
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- During life cycle software is produced early which facilitates customer evaluation and feedback.

Cons of Iterative Model

- More resources may be required.
- Although cost of change is lesser but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which is a risk.
- Highly skilled resources are required for risk analysis.
- Project's progress is highly dependent upon the risk analysis phase.

Rapid Action Development

The RAD (Rapid Application Development) model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product.

Rapid Application development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.

What is RAD?

Rapid application development (RAD) is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product.

In RAD model the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery.

Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process. RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype.

The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable.

RAD Model Design

RAD model distributes the analysis, design, build, and test phases into a series of short, iterative development cycles. Following are the phases of RAD Model:

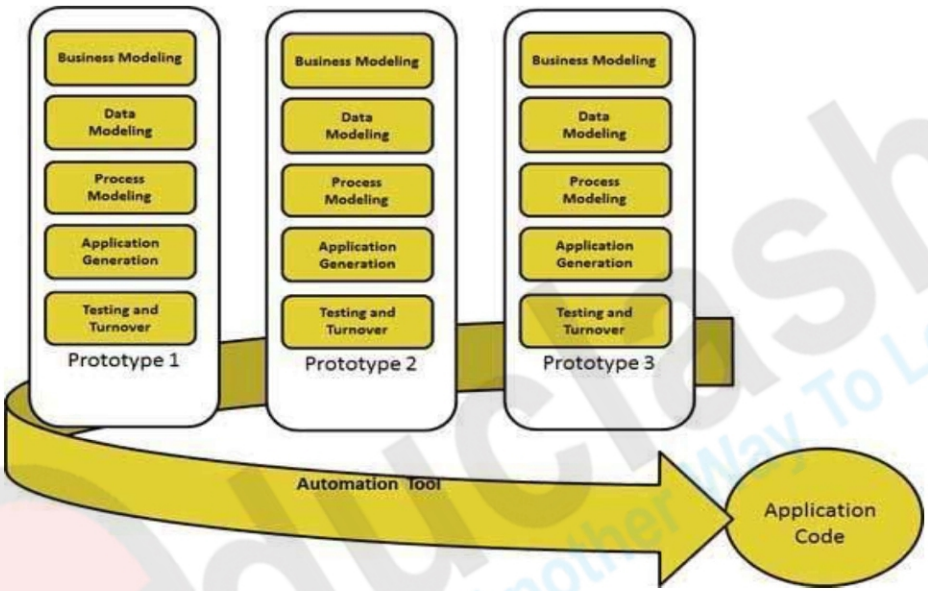
- **Business Modeling:** The business model for the product under development is designed in terms of flow of information and the distribution of information between various business channels. A complete business analysis is performed to find the vital information for business, how it can be obtained, how and when is the information processed and what are the factors driving successful flow of information.

- **Data Modeling:** The information gathered in the Business Modeling phase is reviewed and analyzed to form sets of data objects vital for the business. The attributes of all data sets is identified and defined. The relation between these data objects are established and defined in detail in relevance to the business model.

- **Process Modeling:** The data object sets defined in the Data Modeling phase are converted to establish the business information flow needed to achieve specific business objectives as per the business model. The process model for any changes or enhancements to the data object sets is defined in this phase. Process descriptions for adding , deleting, retrieving or modifying a data object are given.

- **Application Generation:** The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.

- **Testing and Turnover:**The overall testing time is reduced in RAD model as the prototypes are independently tested during every iteration. However the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues.



Pros of RAD Model:

- Changing requirements can be accommodated.
- Progress can be measured.
- Iteration time can be short with use of powerful RAD tools.
- Productivity with fewer people in short time.
- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.

Cons of RAD Model:

- Dependency on technically strong team members for identifying business requirements.
- Only system that can be modularized can be built using RAD.
- Requires highly skilled developers/designers.
- High dependency on modeling skills.

- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.
- Management complexity is more.
- Suitable for systems that are component based and scalable.
- Requires user involvement throughout the life cycle.
- Suitable for project requiring shorter development times.

JAD Model:

Joint Application Development (JAD) is a user requirements elicitation process that involves the system owner and end users in the design and development of an application through a succession of collaborative workshops called JAD sessions. The JAD approach leads to shorter development lifecycles and greater client satisfaction because it draws users and information systems analysts together to jointly design systems in facilitated group sessions. JAD can also be adapted to developing business processes that do not involve computer automation.

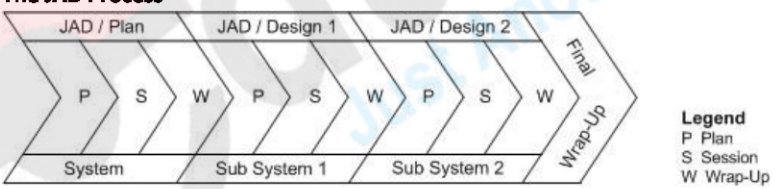
The outcome of a series of JAD sessions is a precise statement of user requirements. A prototype may also be developed.

JAD and Agile

Many of the principles originating from the JAD process have been incorporated into the more recent Agile development process. Where JAD emphasises producing a requirements specification and optionally a prototype, Agile focusses on iterative development, where both requirements and the end software product evolve through regular collaboration between self-organizing cross-functional teams.

JAD remains the best approach if the system must be conceived and delivered in its entirety. That is, it is not possible to deliver the system in small increments of functionality.

The JAD Process



The JAD process proceeds through a number of phases each succeeding phase considering different aspects of the system in ever increasing detail.

JAD/Plan Phase

The JAD/Plan phase is conducted when individuals within an organisation have come up with an idea for a system. They have acquired a sponsor and generated enough support for project funds to have been allocated. Staff have also been committed to the planning activity.

The objective of this phase is to identify the system's high-level requirements and scope. The remainder of the phases deal with fleshing out the various functional areas of the system in greater detail.

Phase Activities

Each phase has three main activities. Planning, conducting the session and wrapping up.

Planning the JAD/Plan Phase

This activity can be a one to five day session conducted with the project sponsor. Activities include:

- Identify high-level requirements
- Define the system scope (business functions, organisational groups, users, geographical locations,external interfaces)
- Estimate the number of sessions required
- Plan requirements elicitation sessions
- Prepare visuals
- Set up document environment
- Setup prototype environment if applicable.

Planning JAD/Design Phases

Organisation Tasks

- Identify the business processes to be addressed
- Identify session participants
- Schedule the session
- Create the agenda/session plan
- Distribute the session plan
- Confirm participants
- Identify participant training needs (e.g. is a JAD process overview required?)

Prepare Session Visuals an Materials

- Session overview
- Requirements capture process orientation
- System context diagram
- High-level requirements overview
- Issues list
- High-level process flow diagrams
- Complex processing descriptions that must be prepared off-line
- Standard forms such as data definition, process description, report/screen layout, interface descriptions

Conducting the JAD Session

Session activities may include:

- Task briefing. Provide a clear statement of what the session is to achieve
- Problem analysis. Reach a complete understanding of the problem
- Product description. Describe the external behaviour of the target system from the user's point of view
- Evaluation. Establish if the requirements are complete correct, consistent and implementable
- Summary. Summarise the requirements discovered

Wrapping-Up the JAD Session

The goal of the wrap-up is to consolidate all the outputs from the session and to present them to the participants and management for approval. Tasks may include:

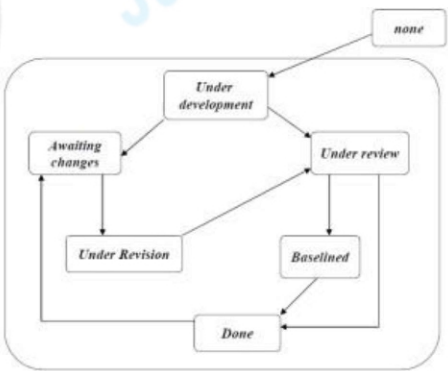
- Complete the JAD/Design document. This can be a requirements specification document and/or various data and process models
- Update the prototype if applicable
- Prepare a sponsor/user presentation
- Conduct a sponsor/user presentation
- Correct JAD outputs if necessary
- Obtain sponsor approval

The final wrap-up involves consolidating all outputs from all sessions into a coherent business requirements specification and a prototype if applicable.

Concurrent Development Model

The concurrent development model - called concurrent engineering.It provides an accurate state of the current state of a project.

- Focus on concurrent engineering activities in a software engineering process such as prototyping, analysis modeling, requirements specification and design.
- Represented schematically as a series of major technical activities, tasks and their associated states.
- Defined as a series of events that trigger transitions from state to state for each of the software engineering activities.
- Often used as the paradigm for the development of client/server applications. A client/server system is composed of a set of functional components. When applied to client/server, the concurrent process model defines activities in two dimensions :(i) System dimension -System level issues are addressed using three activities: design, assembly, and use.(ii) The component dimension is addressed with two activities: design and realization.



Two ways to achieve the concurrency:

1. System and component activities occur simultaneously and can be modeling using the state-oriented approach

2. A typical client/server application is implemented with many components,each can be designed and realized concurrently.

The concurrent process model is applicable to all types of software development and provides an accurate picture of the current state of a project. Rather than confining software engineering activities to a sequence of events, it defines a network of activities. Each activity on the network exists simultaneously with other activities. Events generated within a given activity or at some other place in the activity network trigger transitions among the states of an activity.