

Software Process

- A software process is the set of activities and associated outcome that produce a software product.
- Software engineers mostly carry out these activities.
- These are four key process activities, which are common to all software processes.
- Software processes in software engineering refer to the methods and techniques used to develop and maintain software.

Contd...

- **Software specifications:** The functionality of the software and constraints on its operation must be defined.
- **Software development:** The software to meet the requirement must be produced.
- **Software validation:** The software must be validated to ensure that it does what the customer wants.
- **Software evolution:** The software must evolve to meet changing client needs.

Software Process Model

- A software process model is an abstraction of the actual process, which is being described.
- It can also be defined as a simplified representation of a software process.
- Each model represents a process from a specific perspective.
- Software processes in software engineering refer to the methods and techniques used to develop and maintain software

Type Of Software Process Models

- **A workflow Model –**

It is the sequential series of tasks and decisions that make up a business process.

- **The Waterfall Model –**

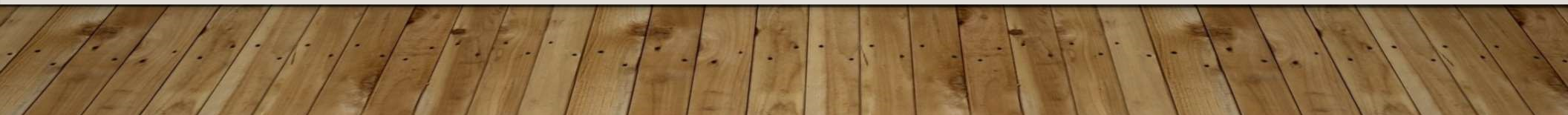
It is a sequential design process in which progress is seen as flowing steadily downwards.

Phases in waterfall model:

- (i) Requirements Specification
- (ii) Software Design
- (iii) Implementation
- (iv) Testing

- **Dataflow Model –**

It is diagrammatic representation of the flow and exchange of information within a system.



Contd...

- **Evolutionary Development Model –**

Following activities are considered in this method:

- (i) Specification
- (ii) Development
- (iii) Validation

- **Role / Action Model –**

Roles of the people involved in the software process and the activities.

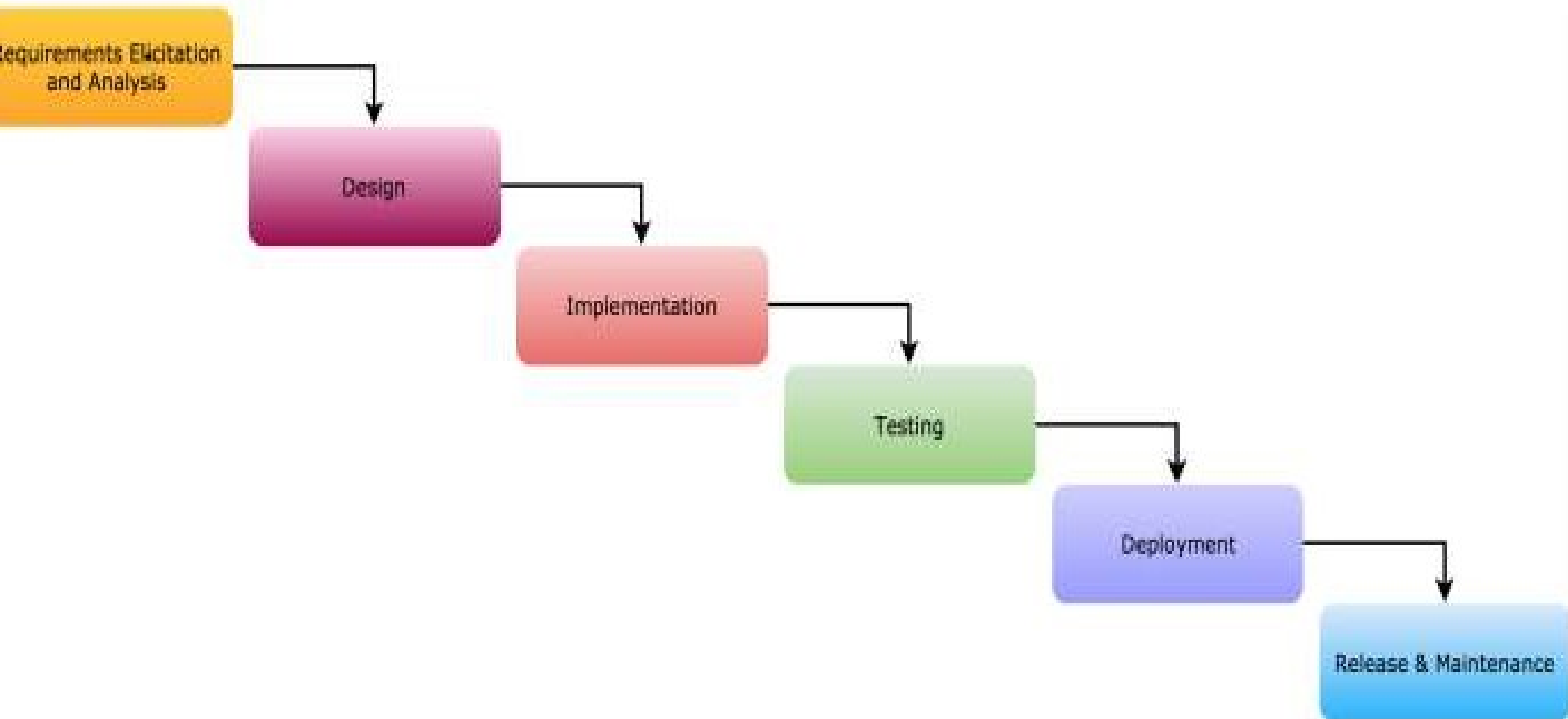
CONTD...

There are various Software development models or methodologies. They are as follows:

- Waterfall model
- V model
- Incremental model
- RAD model
- Agile model
- Iterative model
- Spiral model
- Prototype model

Waterfall Model

- The whole process of software development is divided into various phases.
- It is also referred to as a **linear-sequential life cycle model**.
- Each phase must be completed before the next phase can begin and there is no overlapping in the phases.
- In this model, typically, the outcome of one phase acts as the input for the next phase sequentially.
- The waterfall model is a continuous software development model in which development is seen as flowing steadily downwards (like a waterfall) through the steps of requirements analysis, design, implementation, testing (validation), integration, and maintenance.



Phases In Waterfall Model

Requirement Gathering and analysis:

- All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- It describes the "what" of the system to be produced and not "how."
- In this phase, a large document called **Software Requirement Specification (SRS)** document is created which contained a detailed description of what the system will do in the common language.

CONTD...

System Design:

- The requirement specifications from first phase are studied in this phase and the system design is prepared.
- This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language.
- It defines the overall software architecture together with high level and detailed design. All this work is documented as a Software Design Document (SDD).

Contd...

Implementation:

- With inputs from the 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.
- During this phase, design is implemented. If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD.

CONTD...

Integration and Testing:

- This phase is highly crucial as the quality of the end product is determined by the effectiveness of the testing carried out.
- The better output will lead to satisfied customers, lower maintenance costs, and accurate results. Unit testing determines the efficiency of individual modules.
- All the units developed in the implementation phase are integrated into a system after testing of each unit.
- The modules are tested for their interactions with each other and with the system.
- Post integration the entire system is tested for any faults and failures.

Contd...

- **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.

Waterfall Model Is Most Appropriate

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.

Advantages

- 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.
- Process and results are well documented.

Disadvantages

- 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.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.

Evolutionary Model

- **Evolutionary model** is also referred to as the **successive versions model** and sometimes as the **incremental model**.
- Evolutionary model is a combination of Iterative and Incremental model of software development life cycle.
- In Evolutionary model, the software requirement is first broken down into several modules (or functional units) that can be incrementally constructed and delivered
- The development first develops the **core modules** of the system.
- The initial product skeleton is refined into increasing levels of capability by adding new functionalities in successive versions.
- Each evolutionary model may be developed using an iterative waterfall model of development.

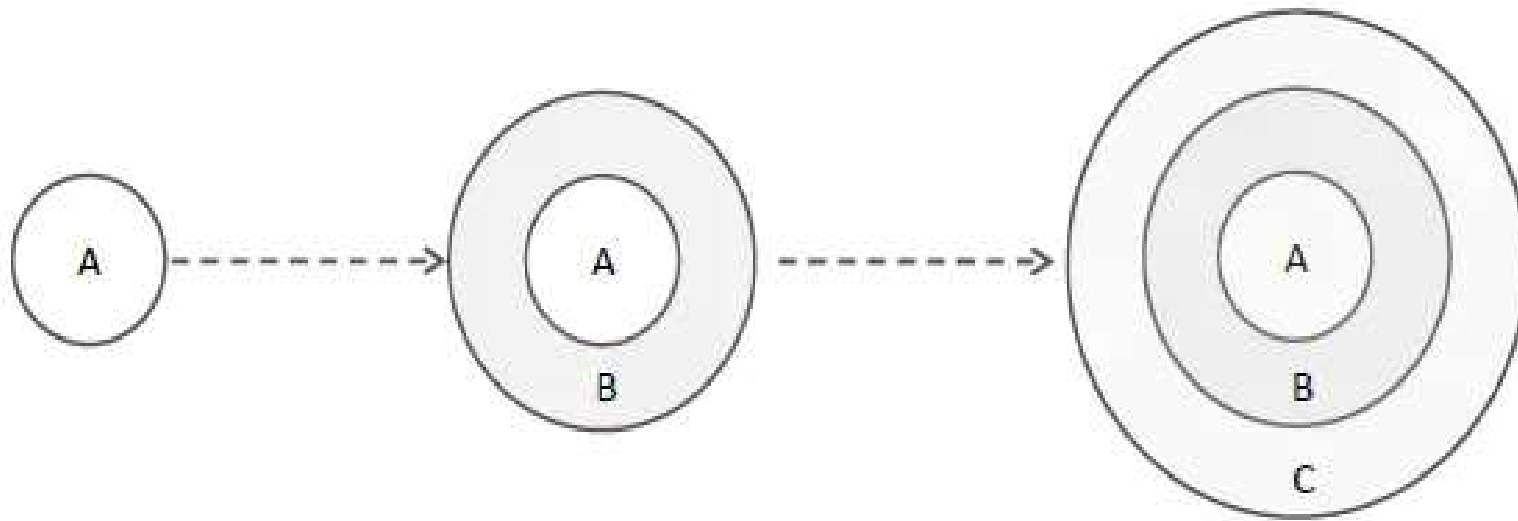


Figure 5: Evolutionary Development of a Software Product

- The **evolutionary model** is shown in Figure 6. Each successive version/model of the product is a fully functioning software capable of performing more work than the previous versions/model.
- The **evolutionary model** is normally useful for very large products, where it is easier to find modules for incremental implementation.

Advantages Of Evolutionary Model

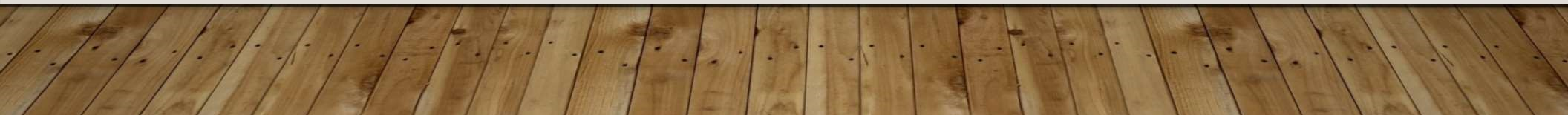
- **Large project:** Evolutionary model is normally useful for very large products.
- User gets a **chance to experiment with a partially developed software** much before the complete version of the system is released.
- Evolutionary model helps to accurately **elicit user requirements** during the delivery of different versions of the software.
- The core modules get tested thoroughly, thereby **reducing the chances of errors** in the core modules of the final products.
- Evolutionary model **avoids the need to commit large resources** in one go for development of the system.

Disadvantages Of Evolutionary Model

- **Difficult to divide the problem into several versions** that would be acceptable to the customer and which can be incrementally implemented and delivered.

Component-based Software Engineering (CBSE)

- It is a process that focuses on the design and development of computer-based systems with the use of reusable software components.
- It emerged from the failure of object-oriented development to support effective reuse.
- Single object classes are too detailed and specific.
- It not only identifies candidate components but also qualifies each component's interface, adapts components to remove architectural mismatches, assembles components into a selected architectural style, and updates components as requirements for the system change.
- The process model for component-based software engineering occurs concurrently with *component-based development*.



COMPONENT-BASED DEVELOPMENT:

- *Component-based development (CBD)* is a CBSE activity that occurs in parallel with domain engineering.
- Using analysis and architectural design methods, the software team refines an architectural style that is appropriate for the analysis model created for the application to be built.

CBSE Essentials

- Independent components specified by their interfaces.
- Component standards to facilitate component integration.
- Middleware that provides support for component interoperability.
- A development process that is geared to reuse.

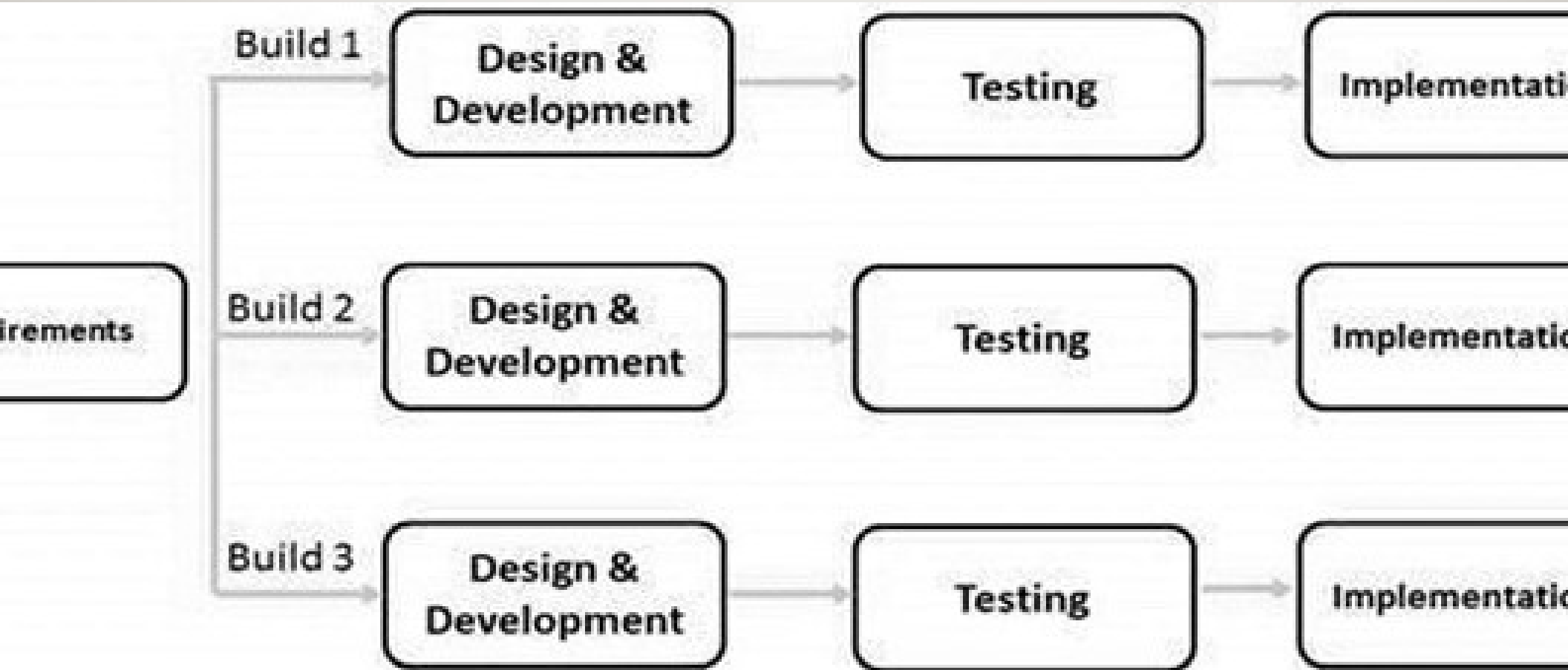
Incremental(Iterative) Model - Design

- Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle.
- 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.
- In this model, each module goes through the requirements, design, implementation and testing phases.
- Every subsequent release of the module adds function to the previous release.
- The process continues until the complete system achieved.

Contd...

- The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).
- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.

The following illustration is a representation of the Iterative and Incremental model –



the various phases of incremental model are as follows:

Requirement analysis:

- In the first phase of the incremental model, the product analysis expertise identifies the requirements.
- And the system functional requirements are understood by the requirement analysis team.
- To develop the software under the incremental model, this phase performs a crucial role.

Design & Development:

- In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success.
- When software develops new practicality, the incremental model uses style and development phase.

Testing:

- In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality.
- In the testing phase, the various methods are used to test the behavior of each task.



4. Implementation:

- Implementation phase enables the coding phase of the development system.
- It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase.
- After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

When We Use The Incremental Model?

- When the requirements are superior.
- A project has a lengthy development schedule.
- When Software team are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.

Advantage Of Incremental Model

- Errors are easy to be recognized.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

Disadvantage of Incremental Model

- Need for good planning
- Total Cost is high.
- Well defined module interfaces are needed.

Spiral Model

- It combines the idea of iterative development with the systematic, controlled aspects of the waterfall model.
- It provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops
- Using the spiral model, the software is developed in a series of incremental releases
- It is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis.
- It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

CONTD...

- It is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model.
- 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 phase 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.

Contd...

Design

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

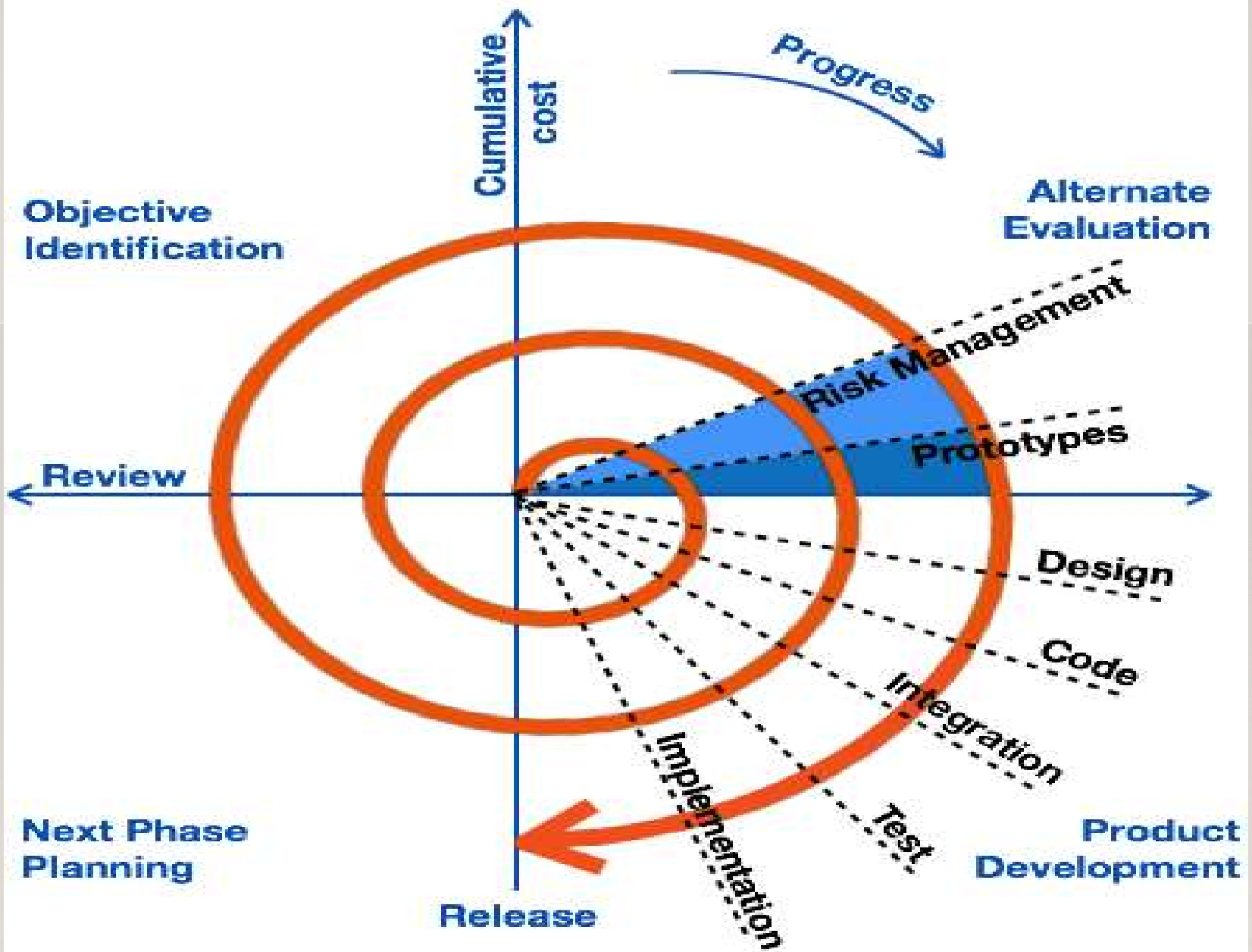
Construct or Build

- The 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.



Evaluation And Risk Analysis

- Risk Analysis includes identifying, estimating and monitoring the 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.



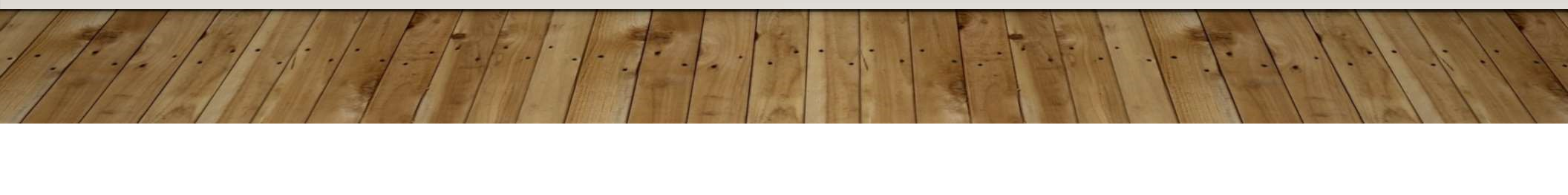
When To Use Spiral Model?

- When deliverance is required to be frequent.
- When the project is large
- When requirements are unclear and complex
- When changes may require at any time
- Large and high budget projects

Advantages

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

Disadvantages

- Management is more complex. Risk analysis needed highly particular expertise.
 - Not suitable for small or low risk projects and could be expensive for small projects.
 - Process is complex. Spiral may go on indefinitely.
 - Large number of intermediate stages requires excessive documentation.
- 

AGILE MODEL

- It refers to a software development approach based on iterative development.
- Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning.
- Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.
- Agile Methods break the product into small incremental builds.
- 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.

CONTD...

- Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks.
- These builds are provided in iterations. Each iteration typically lasts from about one to three weeks.
- The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements.
- Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client

CONTD...

- Every iteration involves cross functional teams working simultaneously on various areas.
- Following are the phases in the Agile model are as follows:
 - Requirements gathering
 - Design the requirements
 - Construction/ iteration
 - Testing/ Quality assurance
 - Deployment
 - Feedback
- At the end of the iteration, a working product is displayed to the customer and important stakeholders.

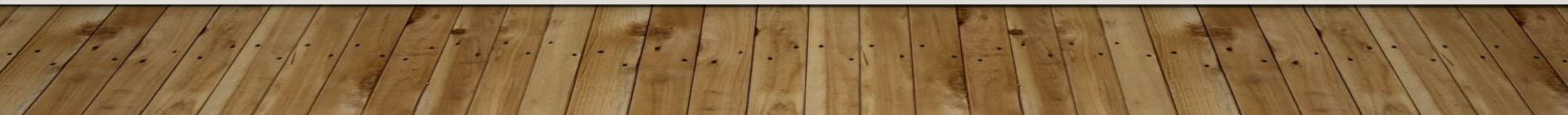
Requirements gathering: In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

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

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

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

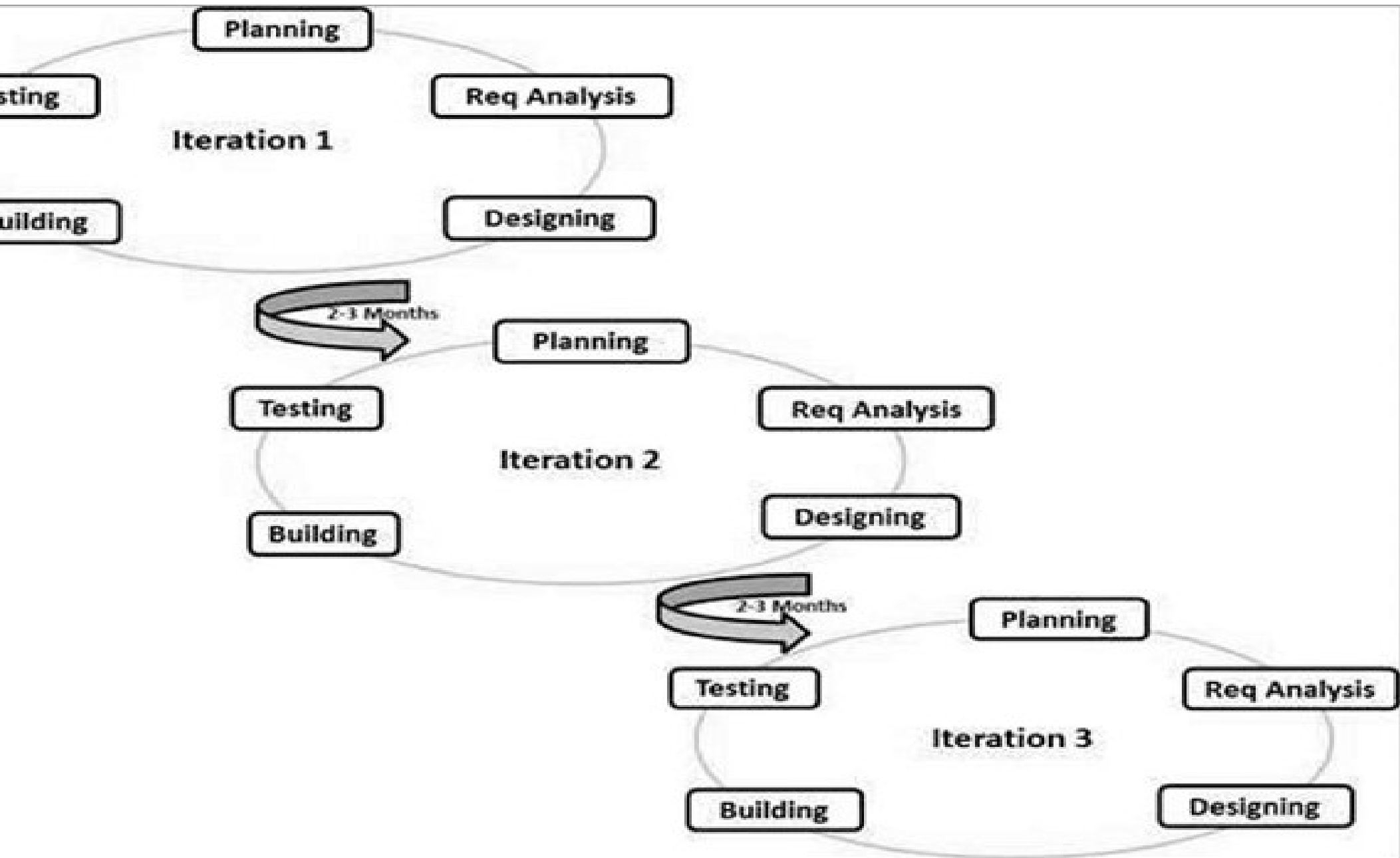
Deployment: In this phase, the team issues a product for the user's work environment.



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

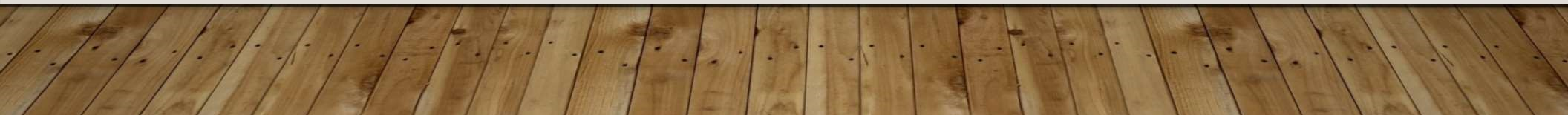


Fig. Agile Model



Following Are The Agile Manifesto Principles

- **Individuals and interactions** – In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
- **Working software** – Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
- **Customer collaboration** – As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
- **Responding to change** – Agile Development is focused on quick responses to change and continuous development.



Advantages

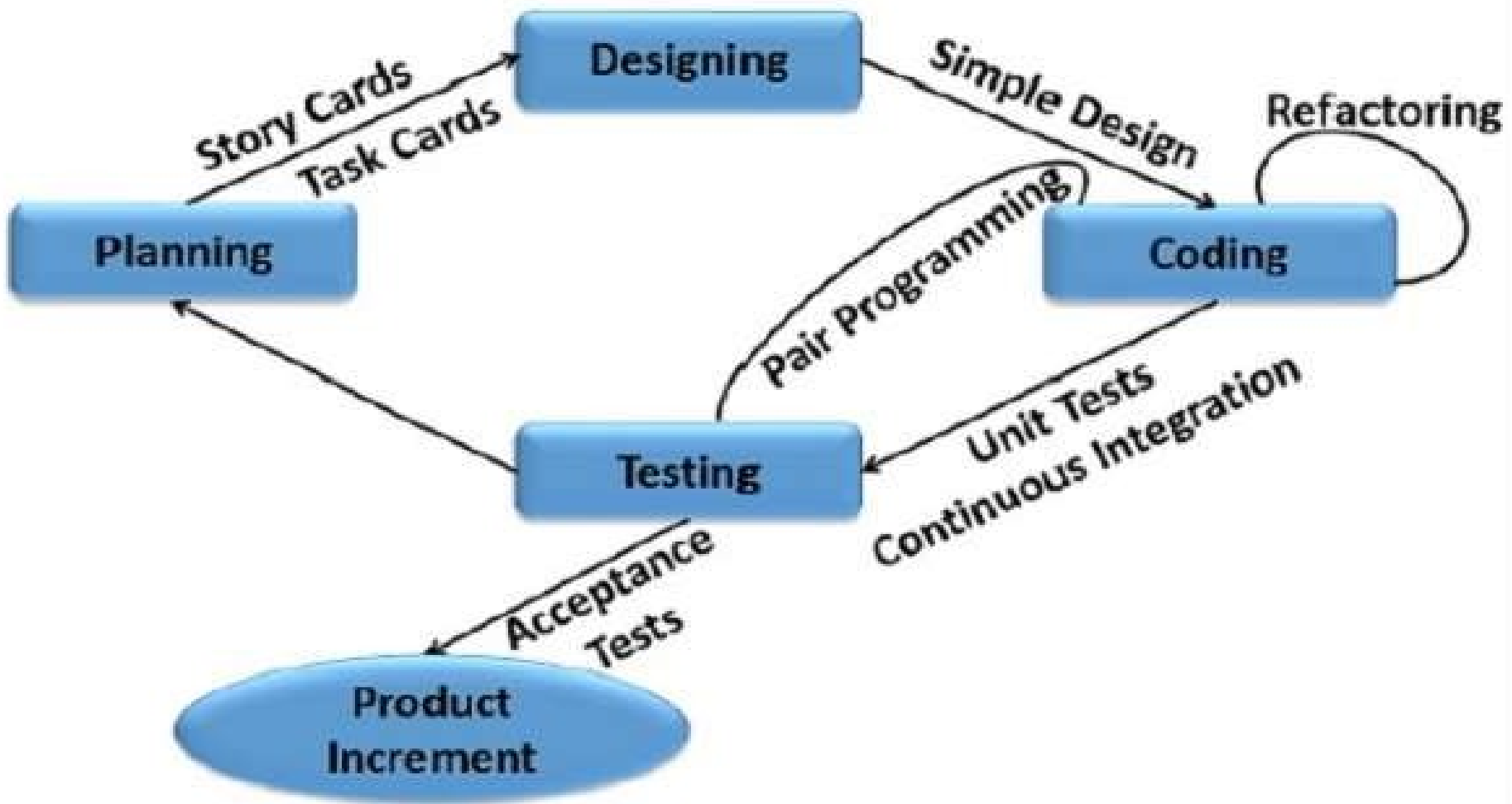
- Is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Easy to manage.
- Gives flexibility to developers.

Disadvantages

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- There is a very high individual dependency, since there is minimum documentation generated.

Extreme Programming (XP)

- Extreme Programming (XP) is an Agile software development methodology that focuses on delivering high-quality software through frequent and continuous feedback, collaboration, and adaptation.
- XP emphasizes a close working relationship between the development team, the customer, and stakeholders, with an emphasis on rapid, iterative development and deployment.
- The process focuses on shorter development cycles to enhance overall software productivity and establishes crucial checkpoints while adapting to new customer requirements.
- Extreme programming was the first Agile software development framework that could overhaul the conventional waterfall model.



Phases of Extreme Programming

I. Planning

- This is the first phase of the Extreme Programming life cycle, which involves user stories along with iterations.
- The team of developers and outline their requirements through user stories that reveal the desired result.
- The team then uses these client requirements to create multiple iterations necessary to achieve the desired functionality on a part-by-part basis.
- In situations where user stories are challenging to estimate and develop iterations, 'spikes' are introduced, which signify the necessity for further research.

2. Designing

- Designing is a subpart of the planning phase.
- It covers one of the principle XP values: 'simplicity'.
- This implies that a good quality design gives definitive logic and structure to system implementation and reduces unnecessary complexities and redundancies.



Coding

- Coding is the most critical phase of all the lifecycle stages.
- The code is written by following coding standards that fall under specific XP practices such as pair programming, metaphor, regular integration, refactoring, code review with collective code ownership, and so on.

Testing

- Testing code helps to remove errors and improves its reliability.
- It suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach test cases are written even before any code is written.
- Alternatively, there is an approval test run at the end of the coding to deliver the client with approval results.



Listening

- The listening phase applies to every development stage.
- It implies maintaining constant communication and feedback with the clients to ensure that their expectations and requirements are always met.
- Besides listening to client requirements, developers can also input the technical aspects of refining the overall system by altering certain software features.



RAD (Rapid Application Development) Model

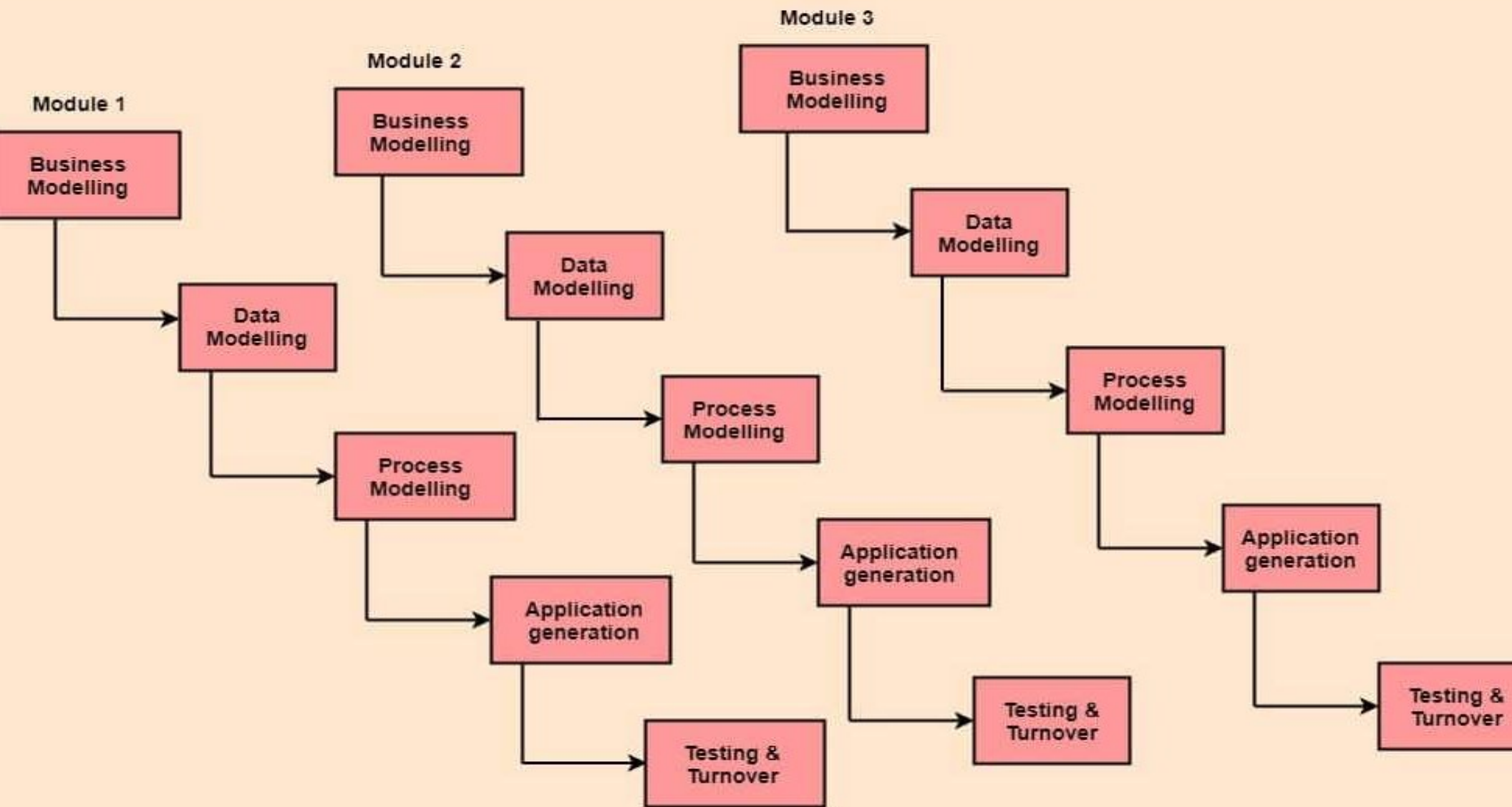
- RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element based construction approach.
- If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a concise time period.
- The RAD 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.

RAD is a concept that products can be developed faster and of higher quality through:

- Gathering requirements using workshops or focus groups
- Prototyping and early, reiterative user testing of designs
- The re-use of software components
- A rigidly paced schedule that refers design improvements to the next product version
- Less formality in reviews and other team communication



Fig: RAD Model



The Various Phases Of RAD Are As Follows

I. Business Modelling:

- 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.
- The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.
- 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 Modelling:

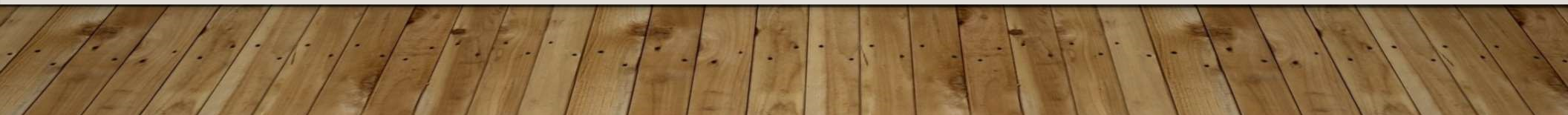
- The data collected from business modeling is refined into a set of data objects (entities) that needed to support the business.
- The attributes of all data sets is identified and defined.
- The relation between these data objects are established and defined in detail in relevance to business model.

. Process Modelling:

- The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function.
- The process model for any changes or enhancements to the data object sets is defined in this phase
- Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

. Application Generation:

- Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.
- The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes



5. Testing & Turnover:

- Many of the programming components have already been tested since RAD emphasis reuse.
- This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised.

Advantages

- 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 a short time.
- Reduced development time.
- Increases reusability of components.
- Encourages customer feedback.
- Integration from very beginning solves a lot of integration issues.

Disadvantages

- 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 Modelling skills.
- Management complexity is more.
- Suitable for systems that are component based and scalable.
- Requires user involvement throughout the life cycle.

Software Prototyping

- The Software Prototyping refers to building software application prototypes which displays 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.
- Prototyping is used to allow the users evaluate developer proposals and try them out before implementation.



Steps Of Prototype Model

Basic requirement identification:

- This step involves understanding the basic product requirements in terms of the user interface.
- The software publisher decides the functionality, who the user will likely be, and what the user will want from the product.

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.
- While, the workarounds are used to give the same look and feel to the customer in the prototype developed.

Review 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.

Revise 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 the actual implementation.
- The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until the customer expectations are met.



Types Of Software Prototyping

- There are two main types of prototypes. They are, the *throwaway model* and the *evolutionary model*.

1. Throwaway Prototyping

- The **throwaway model** is designed to be discarded once the review process has been completed.
- It looks at what the end product may look like and it's typically not well defined.

2. Evolutionary Prototyping

- The **evolutionary model** for prototyping is based on building actual functional prototypes with minimal functionality in the beginning.
- By using evolutionary prototyping, the well-understood requirements are included in the prototype and the requirements are added when they are understood.

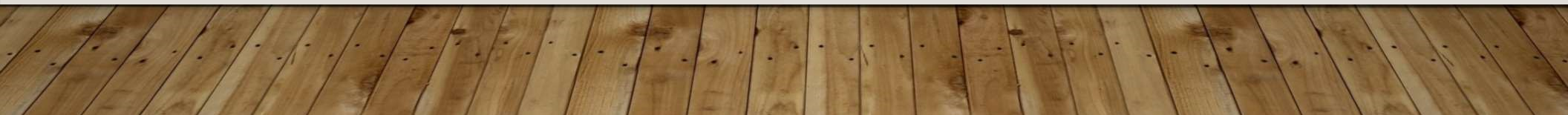


The advantages of software prototyping are:

- It reduces time and cost, as the defects are detected much earlier.
- Quicker feedback from users.
- Increased user involvement in the software product production.
- Missing functionalities are identified easily.

The disadvantages of software prototyping are:

- Users may get confused between the prototypes and actual systems.
- Too much effort may be invested in building prototypes.
- A risk of insufficient requirement analysis due to over-dependency on the prototype.



Rational Unified Process (RUP)

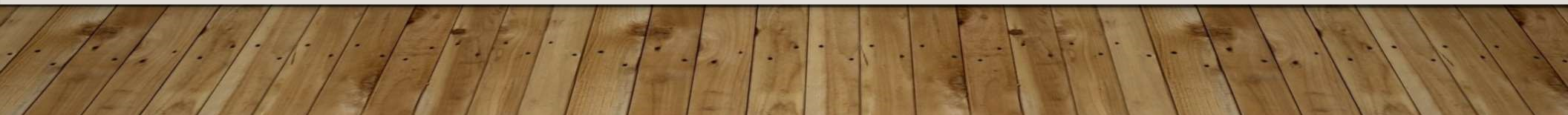
- The rational unified process (RUP) is a software engineering and development process focused on using the unified modeling language (UML) to design and build software.
- Using the RUP process allows us to operate business analysis, design, testing and implementation throughout the software development process and its unique stages, helping us create a customized product.

ception

- Communication and planning are the main ones.
- Identifies the scope of the project using a use-case model allowing managers to estimate costs and time required.
- Customers' requirements are identified and then it becomes easy to make a plan for the project.
- The project plan, Project goal, risks, use-case model, and Project description, are made.
- The project is checked against the milestone criteria and if it couldn't pass these criteria then the project can be either canceled or redesigned.

laboration

- Planning and modeling are the main ones.
- A detailed evaluation and development plan is carried out and diminishes the risks.
- Revise or redefine the use-case model, business case, and risks.
- Again, checked against milestone criteria and if it couldn't pass these criteria then again project can be canceled or redesigned.
- Executable architecture baseline.



Construction –

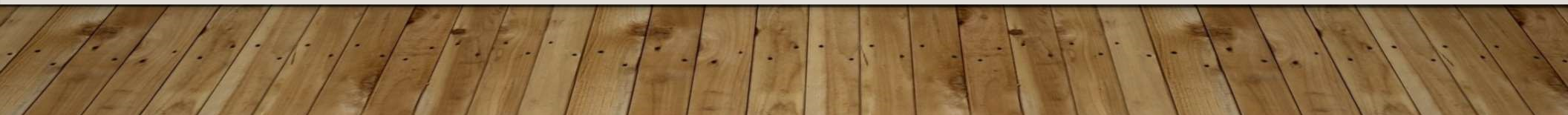
- The project is developed and completed.
- System or source code is created and then testing is done.
- Coding takes place.

Transition –

- The final project is released to the public.
- Transit the project from development into production.
- Update project documentation.
- Beta testing is conducted.
- Defects are removed from the project based on feedback from the public.

Production –

- The final phase of the model.
- The project is maintained and updated accordingly.



Computer-Aided Software Engineering (CASE)

- It is the implementation of computer-facilitated tools and methods in software development.
- It is the domain of software tools used to design and implement applications. CASE tools are similar to and were partly inspired by Computer-Aided Design (CAD) tools used for designing hardware products.
- CASE tools were used for developing high-quality, defect-free, and maintainable software.
- They create a framework for managing projects and are intended to help users stay organized and improve productivity.
- CASE ensures a check-pointed and disciplined approach and helps designers, developers, testers, managers, and others to see the project milestones during development.

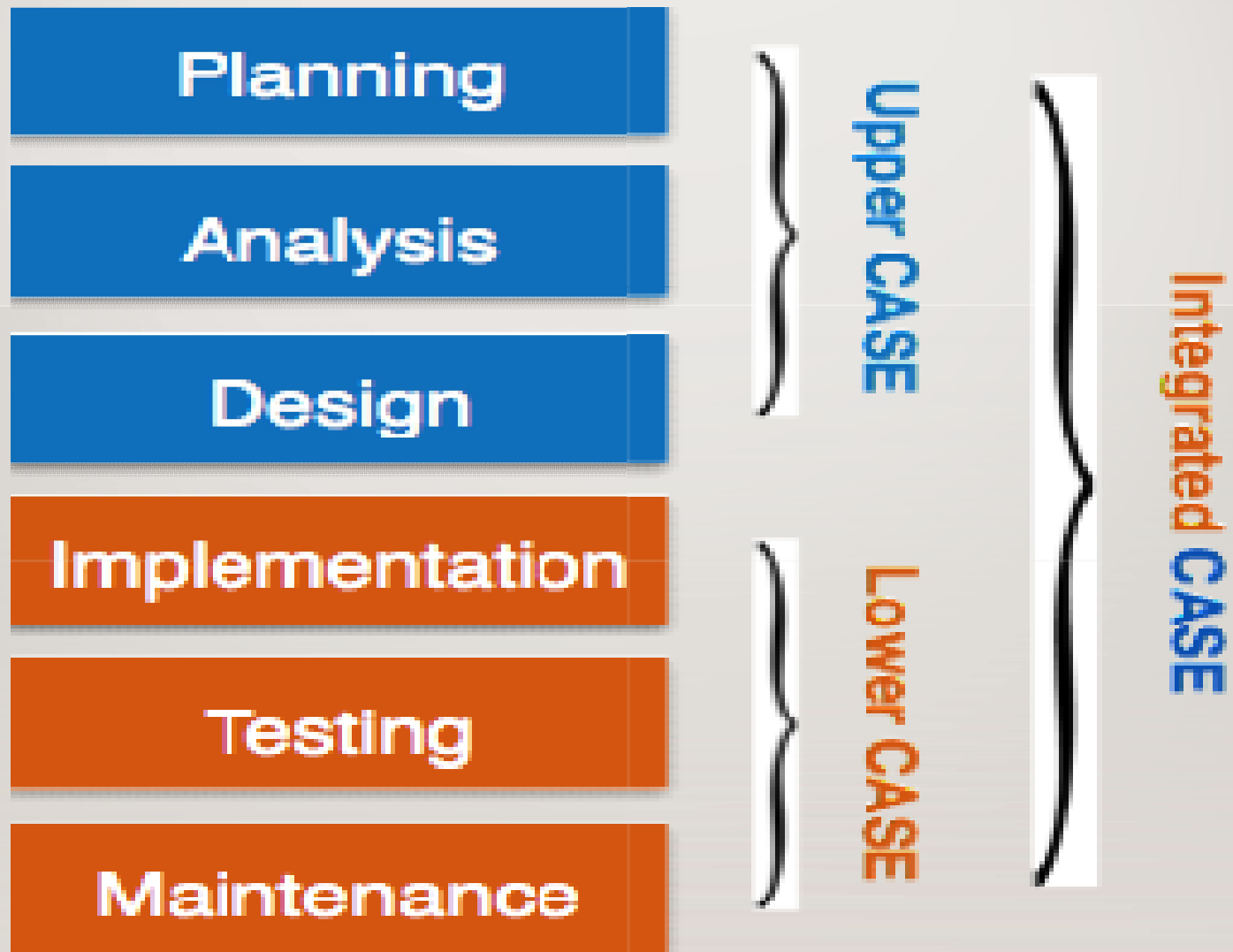
CONTD...

- CASE can also help as a warehouse for documents related to projects, like business plans, requirements, and design specifications.
- One of the major advantages of using CASE is the delivery of the final product, which is more likely to meet real-world requirements as it ensures that customers remain part of the process

COMPONENTS OF CASE TOOLS

Central Repository -

- CASE tools require a central repository, which can serve as a source of common, integrated and consistent information.
- Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored.
- Central repository also serves as data dictionary.



CONTD...

- **Upper Case Tools** - Upper CASE tools are used in planning, analysis and design stages of SDLC.
- **Lower Case Tools** - Lower CASE tools are used in implementation, testing and maintenance.
- **Integrated Case Tools** - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

Case Tools Types

Diagram tools

- These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts.

Process Modeling Tools

- Process modeling is method to create software process model, which is used to develop the software.
- Process modeling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, EPF Composer

Project Management Tools

These tools are used for project planning, cost and effort estimation, project scheduling and resource planning.

Managers have to strictly comply project execution with every mentioned step in software project management.

Project management tools help in storing and sharing project information in real-time throughout the organization. For example, Creative Pro Office, Trac Project, Basecamp.

Documentation Tools

Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project.

It generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc



The end user documents describe the functioning and how-to of the system such as user manual

Analysis Tools

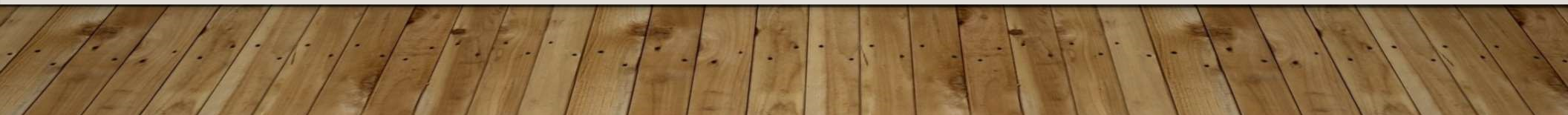
These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions.

For example, Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis.

Design Tools

These tools help software designers to design the block structure of the software, which further be broken down in smaller modules using refinement techniques.

These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design



Configuration Management Tools

An instance of software is released under one version. Configuration Management tools deal

- Version and revision management
- Baseline configuration management
- Change control management

CASE tools help in this by automatic tracking, version management and release management. example, Fossil, Git, Accu REV.

Change Control Tools

These tools are considered as a part of configuration management tools. They deal with changes made to the software after its baseline is fixed or when the software is first released.

CASE tools automate change tracking, file management, code management and more. It also helps in enforcing change policy of the organization.

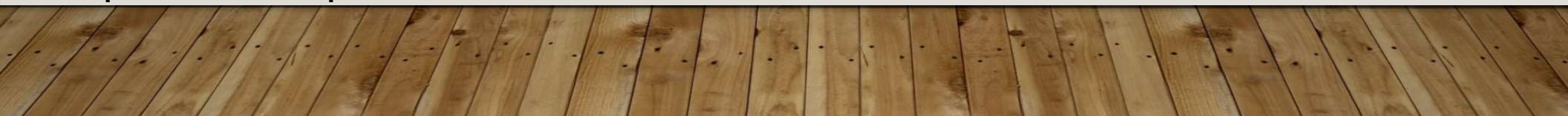


Programming Tools

- These tools consist of programming environments like IDE (Integrated Development Environment), in-built modules library and simulation tools.
- These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse.

Prototyping Tools

- Software prototype is simulated version of the intended software product.
- Prototype provides initial look and feel of the product and simulates few aspect of actual product.
- Prototyping CASE tools essentially come with graphical libraries.
- They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing information.
- In addition, they provide simulation of software prototype. For example, Serena prototype composer, Mockup Builder.



Web Development Tools

These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on.

Web tools also provide live preview of what is being developed and how will it look after completion. For example, Fontello, Adobe Edge Inspect, Foundation 3, Brackets.

Quality Assurance Tools

Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of quality as per organization standards.

QA tools consist of configuration and change control tools and software testing tools. For example, SoapTest, AppsWatch, JMeter.



Maintenance Tools

- Software maintenance includes modifications in the software product after it is delivered.
- Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC.