Ch-2

Software Development Process models

Software Process

- A structured set of activities required to develop a software system.
- A software process (also knows as software methodology) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.
- There are also **supporting activities** such as configuration and change management, quality assurance, project management, user experience.
- When we talk about a process, we usually talk about the activities in it. However, a process also includes the process description, which includes:
  - **Products**: The outcomes of the an activity. For example, the outcome of architectural design maybe a model for the software architecture.
  - **Roles**: The responsibilities of the people involved in the process. For example, the project manager, programmer, etc.
  - **Pre and post conditions**: The conditions that must be true before and after an activity. For example, the pre condition of the architectural design is the requirements have been approved by the customer, while the post condition is the diagrams describing the architectural have been reviewed.

Activities involved in Software Process

- There are four major activities involved in software process:
  - **Software specification** : Define the main functionalities of the software and the constrains around them.(defining what the system should do)
  - S**oftware design and implementation**: The software is to be designed and programmed.(defining the organization of the system and implementing the system;)
  - **Software verification and validation**: The software must conforms to it's specification and meets the customer needs.(checking that it does what the customer wants;)
  - **Software evolution** (software maintenance): The software is being modified to meet customer and market requirements changes.(changing the system in response to changing customer needs.)

Software Process Model

- A software process model is an abstract/simplified representation of a process. It presents a description of a process from some particular perspective
- When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc. and the ordering of these activities.
- Some methodologies are sometimes known as **software development life cycle** (SDLC) methodologies, though this term could also be used more generally to refer to any methodology.

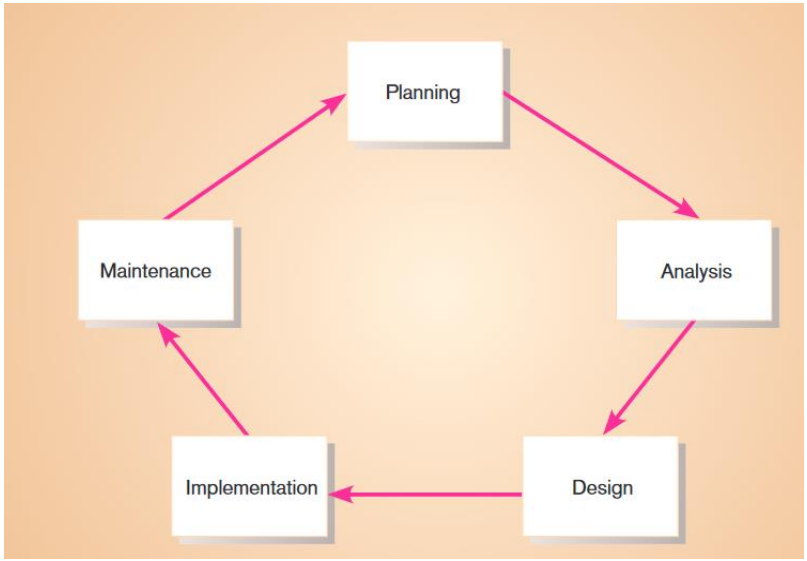


Fig: SDLC Phases

Software process models

- The waterfall model
- Evolutionary Development
- Component Based Software Engineering(CBSE)
- Software Prototyping
- Process Iteration
  - Incremental development
  - Spiral Development
- Rapid Software Development
  - Rapid Application Development(RAD)
  - Agile Methods
  - Extreme Programming

- Rational Unified Process(RUP)
- In practice, most large systems are developed using a process that incorporates elements from all of these models.
- [Note:*Plan-driven process* is a process where all the activities are planned first, and the progress is measured against the plan. While the *agile process*, planning is incremental and it's easier to change the process to reflect requirement changes.]

Sequential(Waterfall Model)

- The first published model of the software development process was Waterfall model. It was proposed by Royce in 1970s.
- Because of the cascade from one phase to another, this model is known as waterfall/software life cycle.
- This model takes the fundamental process activities of specification, development, validation and evolution and represents them as separate phases such as **requirements specification, software design, implementation, testing and maintenance**
- In principle, the result of each phase is one or more documents that should be approved and the next phase shouldn't be started until the previous phase has completely been finished.
- In practice, however, these phases overlap and feed information to each other. For example, during design, problems with requirements can be identified, and during coding, some of the design problems can be found, etc.
- The software process therefore is not a simple linear but involves feedback from one phase to another. So, documents produced in each phase may then have to be modified to reflect the changes made.

The structure/phases of Waterfall model

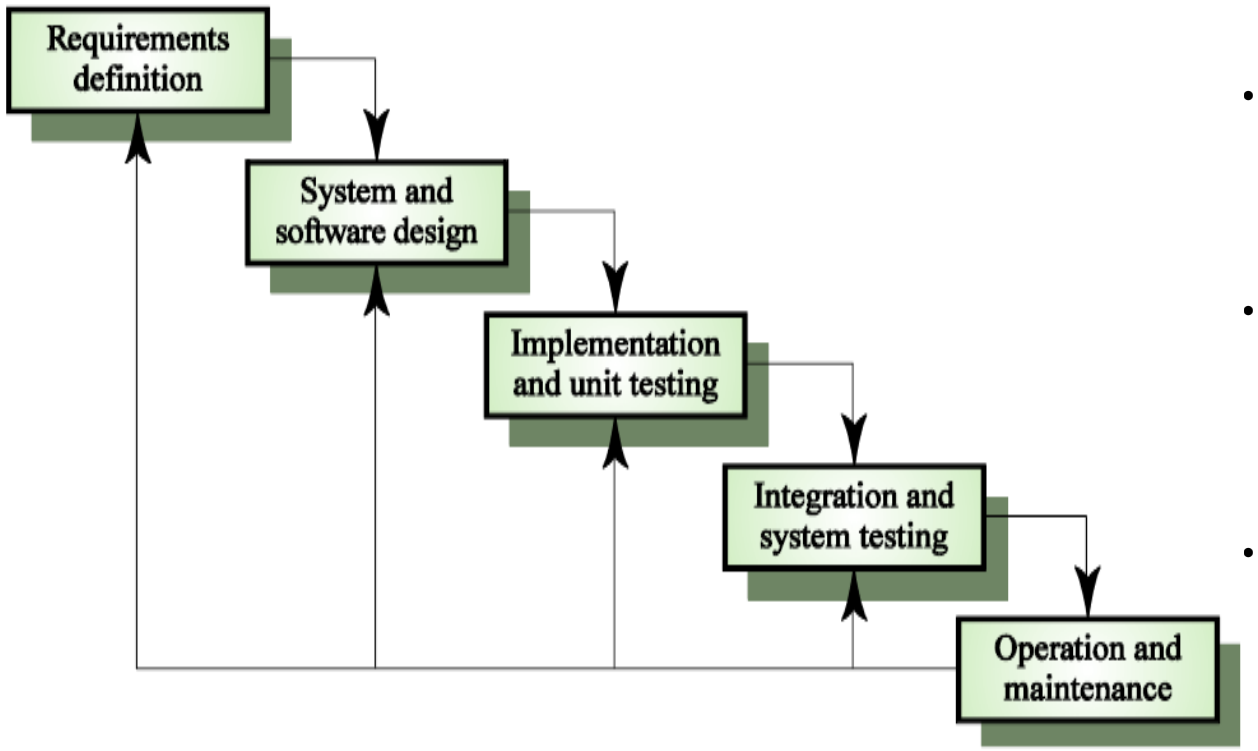


Fig: Waterfall (Sequential) Model

- **Requirements analysis and definition:**
  - The system's service, constraints and goals are established by consultation with system users. They are often defined in detail and serve as a system specification.
- **System and software design**:
  - The system design process partitions the requirements to either hardware or software systems.
  - It establishes overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationship
- **Implementation and unit testing**:
  - During this stage, the software design is realised as asset of programs and program units.
  - Unit testing involves verifying that each unit meets its specification.
- **Integration and system testing**:
  - The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.
- **Operation and maintenance**:
  - This is the longest life-cycle phase. The system is installed and put into practical use.
  - Maintenance involves correcting errors which were not discovered earlier stages of the life-cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered.

When to use?

In principle, the waterfall model should only be applied when requirements are well understood and unlikely to change radically during development as this model has a relatively rigid structure which makes it relatively hard to accommodate change when the process in underway.

Advantages/Strengths

- Easy to understand even by non-technical person, i.e customers.
- Each phase has well defined inputs and outputs.
- Easy to use as software development proceeds,
- Each stage has well defined deliverables.
- Helps the project manager in proper planning of the project.

Disadvantages/weaknesses

- The drawback of the waterfall model is the difficulty of accommodating change after the process is underway because of sequential nature
- Inflexible partitioning of the project into distinct stages
- This makes it difficult to respond to changing customer requirements
- ***This model is only appropriate when the requirements are well-understood***

Evolutionary development Model

- Evolutionary model is based on the idea of developing an initial implementation, exposing to this to user comment and refining it through many versions until an adequate system has been developed.
- Specification, development and validation activities are interleaved rather than separate, with rapid feedback across activities.
- Evolutionary models are iterative. They are characterized in a manner that enables software engineers to develop increasingly more complete versions of the software.
- Evolutionary process model resembles iterative development model.
- The same phases as defined for the waterfall model occur here in a cyclical fashion.
- This model differs from iterative development model in the sense that this does not require a useable product at the end of each cycle.
- Eg. In simple database application one cycle might implement the graphical user interface, another file manipulation, another queries, and another updates. All four cycles must complete before there is working product available.
- In evolutionary development, requirements are implemented by category rather than by priority.
- This model is useful for projects using new technology that is not well understood.
- This is also used for complex projects where all functionality must be delivered at one time, but the requirements are unstable or not well understood at the beginning.

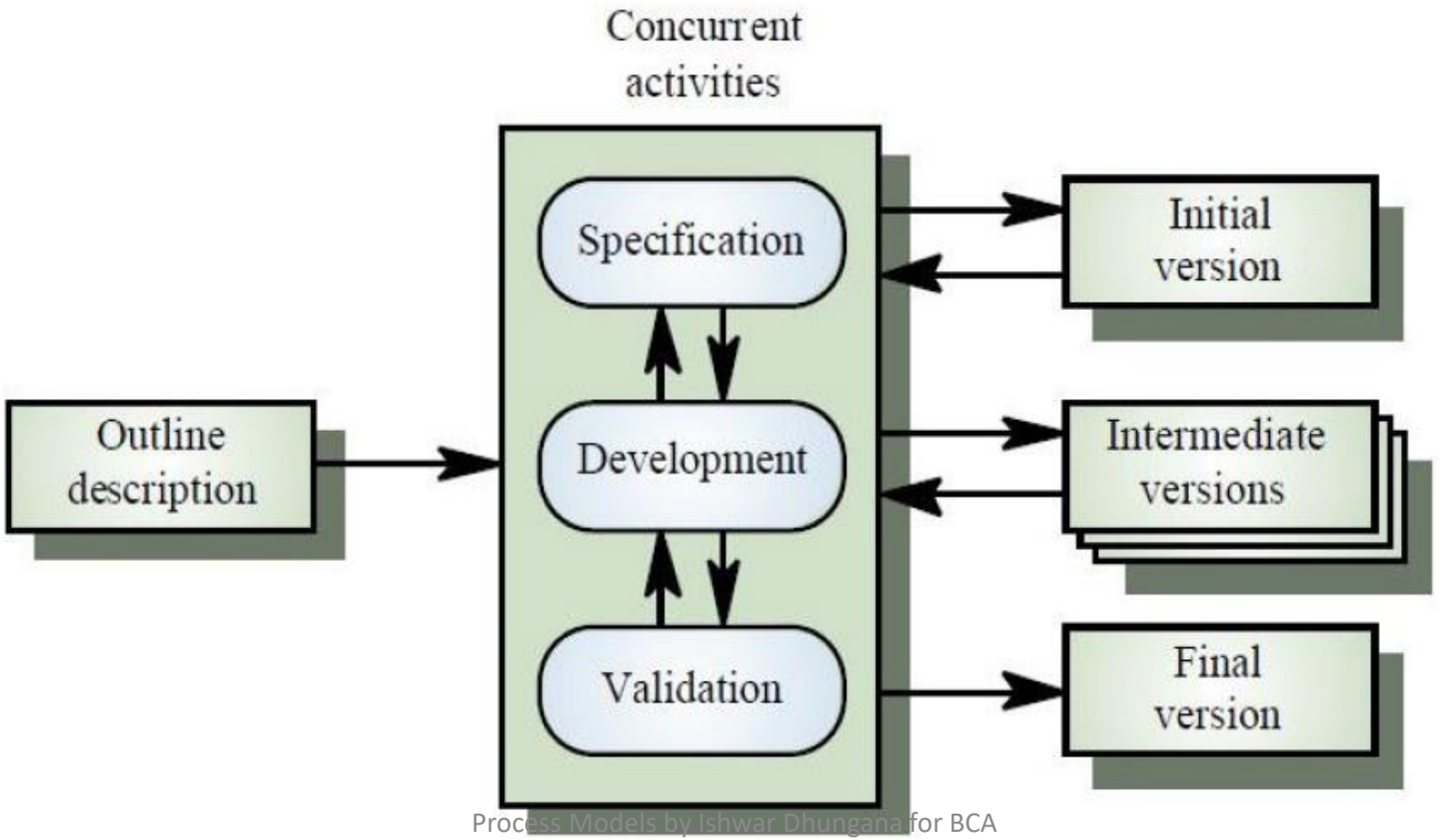
Types:

1. Exploratory development:

Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements

2.Throw-away prototyping:

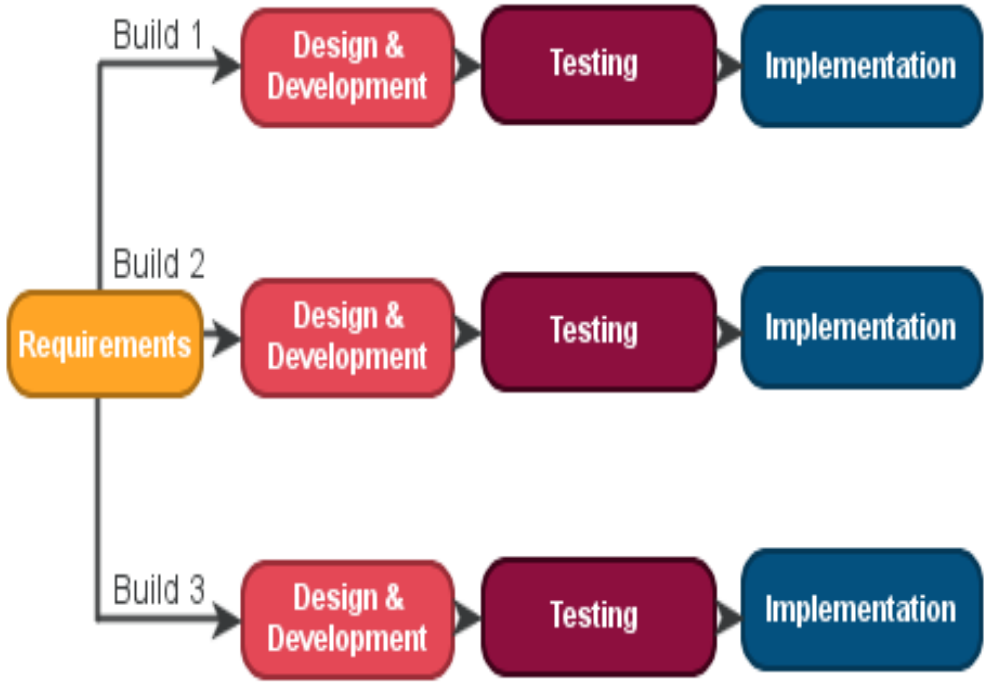
Objective is to understand the system requirements. Should start with poorly understood requirements



Advantages/Disadvantages of evolutionary development

- Advantages
  - An evolutionary approach is often more effective than waterfall approaches in producing systems that meet the immediate needs of customers.
  - The advantage of software process that is based on an evolutionary approach is that the specification can be developed incrementally.
  - As users develop a better understanding of their problem, this can be reflected in the software system.
- Disadvantages
  - The process is not visible: Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
  - Systems are often poorly structured: Continual change tends to corrupt the software structure. Incorporating software changes becomes increasingly difficult and costly.
  - Special Skills may be required

Iterative Development Model



- The process of Iterative Model is **cyclic**, unlike the more traditional models that focus on a **rigorous step-by-step process of development**. In this process, once the initial planning is complete, a handful of phases are **repeated again and again, with the completion of each cycle incrementally improving and iterating on the software**

Phases:

- **Planning Phase:** This is the first stage of the iterative model, where proper planning is done by the team, which helps them in mapping out the specifications documents, establish software or hardware requirements and generally prepare for the upcoming stages of the cycle.
- **Analysis and Design Phase:** Once the planning is complete for the cycle, an analysis is performed to point out the appropriate business logic, database models and to know any other requirements of this particular stage. Moreover, the design stage also occurs in this phase of iterative model, where the technical requirements are established that will be utilized in order to meet the need of analysis stage.
- **Implementation Phase:** This is the third and the most important phase of the iterative model. Here, the actual implementation and coding process is executed. All planning, specification, and design documents up to this point are coded and implemented into this initial iteration of the project.
- **Testing Phase:** After the current build iteration is coded and implemented, testing is initiated in the cycle to identify and locate any potential bugs or issues that may have been in the software.
- **Evaluation Phase:** The final phase of the Iterative life cycle is the evaluation phase, where the entire team along with the client, examine the status of the project and validate whether it is as per the suggested requirements.

When to use Iterative Model?

-When the requirements of the complete system are clearly defined and understood.
-The major requirements are defined, while some functionalities and requested enhancements evolve with the process of the development process.
-A new technology is being used and is being learnt by the development team, while they are working on the project.
-When the resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.

Advantages

- It is easily adaptable to the ever changing needs of the project as well as the client.
- It is best suited for agile organizations.
- It is more cost effective to change the scope or requirements in Iterative model.
- Parallel development can be planned.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed.
- In iterative model less time is spent on documenting and more time is given for designing.
- One can get reliable user feedback, when presenting sketches and blueprints of the product to users for their feedback.

Disadvantages

- More and skilled resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- It is not suitable for smaller projects.
- Project progress is highly dependent upon the risk analysis phase.

Prototyping Model

- A prototype is a version of a system or part of the system that's developed quickly to check the customer's requirements or feasibility of some design decisions.
- So, a prototype is useful when a customer or developer is not sure of the requirements, or of algorithms, efficiency, business rules, response time, etc.
- In prototyping, the client is involved throughout the development process, which increases the likelihood of client acceptance of the final implementation.
- While some prototypes are developed with the expectation that they will be discarded**(throwaway),** it is possible in some cases to evolve from prototype to working system**(evolutionary).**
- A software prototype can be used:
  - In the **requirements engineering**, a prototype can help with the elicitation and validation of system requirements.
  - It allows the users to experiment with the system, and so, refine the requirements. They may get new ideas for requirements, and find areas of strength and weakness in the software.
  - Furthermore, as the prototype is developed, it may reveal errors and in the requirements. The specification maybe then modified to reflect the changes.
  - In the **system design**, a prototype can help to carry out deign experiments to check the feasibility of a proposed design.
- For example, a database design may be prototyped and tested to check it supports efficient data access for the most common user queries.

Prototype model phases

1. Requirements gathering and analysis:

A prototyping model begins with requirements analysis and the requirements of the system are defined in detail.

2. Quick design:

When requirements are known, a preliminary design or quick design for the system is created. A quick design helps in developing the prototype.

3. Build prototype:

Information gathered from quick design is modified to form the first prototype, which represents the working model of the required system.

4. User evaluation:

Next, the proposed system is presented to the user for thorough evaluation of the prototype to recognize its strengths and weaknesses such as what is to be added or removed.

Comments and suggestions are collected from the users and provided to the developer.

5. Refining prototype:

Once the user evaluates the prototype and if he is not satisfied, the current prototype is refined according to the requirements. That is, a new prototype is developed with the additional information provided by the user. The new prototype is evaluated just like the previous prototype.

This process continues until all the requirements specified by the user are met. Once the user is satisfied with the developed prototype, a final system is developed on the basis of the final prototype.

6. Engineer product:

Once the requirements are completely met, the user accepts the final prototype. The final system is evaluated thoroughly followed by the routine maintenance on regular basis for preventing large-scale failures and minimizing downtime.

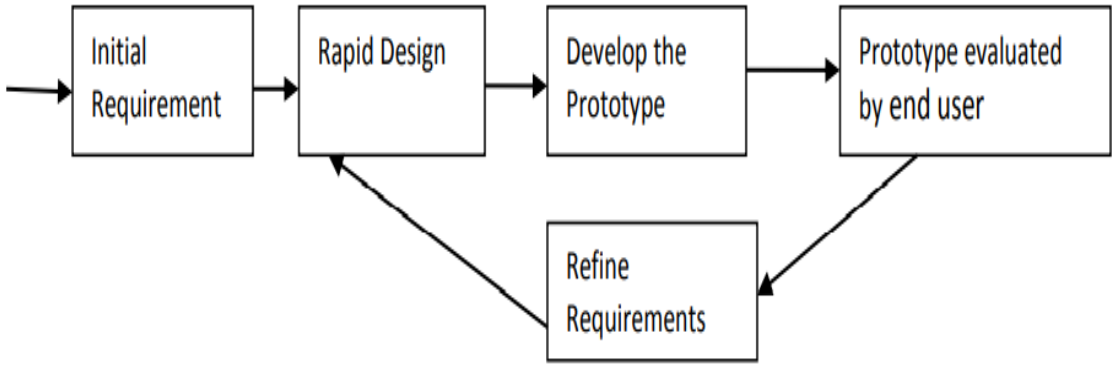
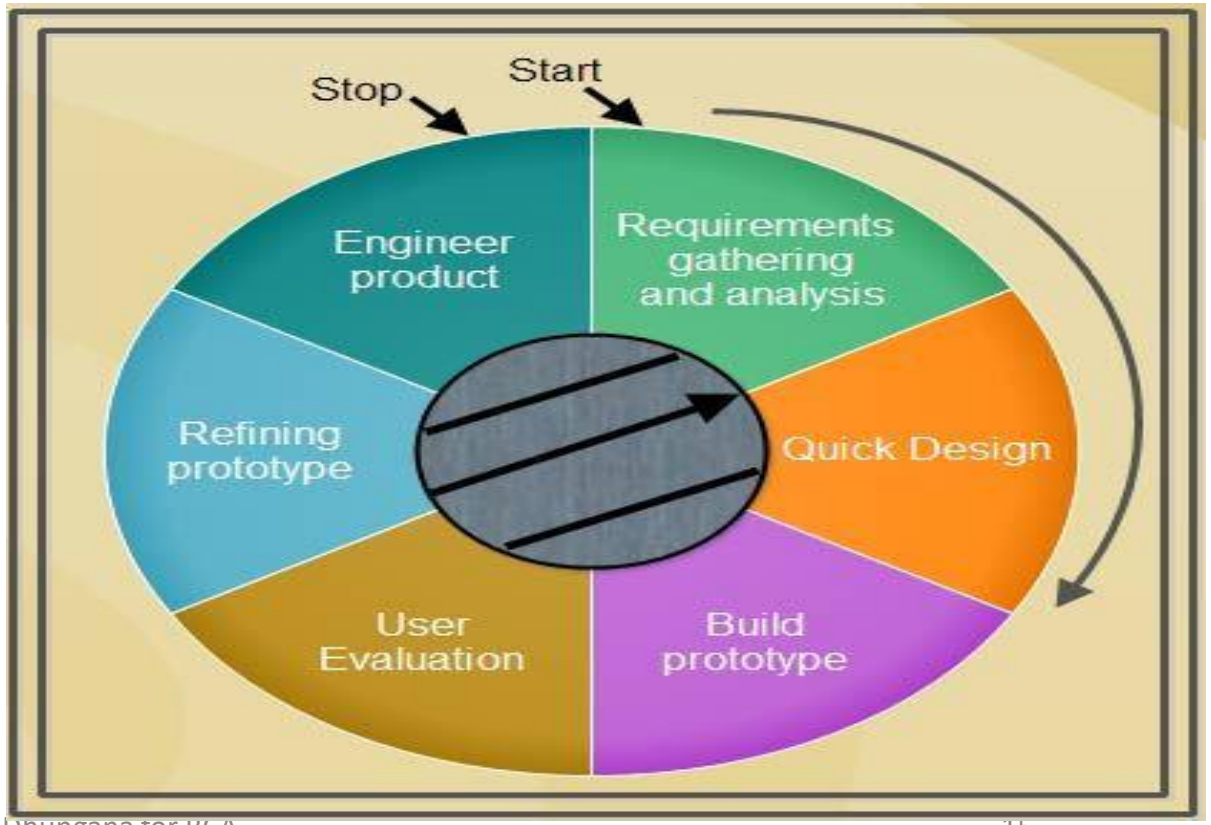


Fig: The Prototype Model



Advantages/Disadvantages

- **Advantages:**

- Reduce the risk of incorrect user requirement
- Good where requirement are changing/uncommitted
- Regular visible process aids management
- Support early product marketing
- Reduce Maintenance cost.
- Errors can be detected much earlier as the system is made side by side.

- **Weaknesses:**

  - This model is time consuming and expensive.
  - The developer loses focus of the real purpose of prototype and hence, may compromise with the quality of the software. For example, developers may use some inefficient algorithms or inappropriate programming languages while developing the prototype.
  - Prototyping can lead to false expectations. For example, a situation may be created where the user believes that the development of the system is finished when it is not.
  - An unstable/badly implemented prototype often becomes the final product.

- Require extensive customer collaboration
  - Costs customer money
  - Needs committed customer
  - Difficult to finish if customer withdraw
  - May be too customer specific, no broad market
- Difficult to know how long the project will last.
- Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback.
- Prototyping tools are expensive.
- Special tools & techniques are required to build a prototype.

Incremental Process model

- The incremental model combines the elements of waterfall model and they are applied in an iterative fashion.
- The first increment in this model is generally a core product.
- Each increment builds the product and submits it to the customer for any suggested modifications.
- The next increment implements on the customer's suggestions and add additional requirements in the previous increment.
- This process is repeated until the product is finished.
  For example, the word-processing software is developed using the incremental model.
- **Advantages of incremental model:**
  - This model is flexible because the cost of development is low and initial product delivery is faster.
  - It is easier to test and debug during the smaller iteration.
  - The working software generates quickly and early during the software life cycle.
  - The customers can respond to its functionalities after every increment.
- **Disadvantages of the incremental model:**
  - The cost of the final product may cross the cost estimated initially.
  - This model requires a very clear and complete planning.
  - The planning of design is required before the whole system is broken into small increments.
  - The demands of customer for the additional functionalities after every increment causes problem during the system architecture.

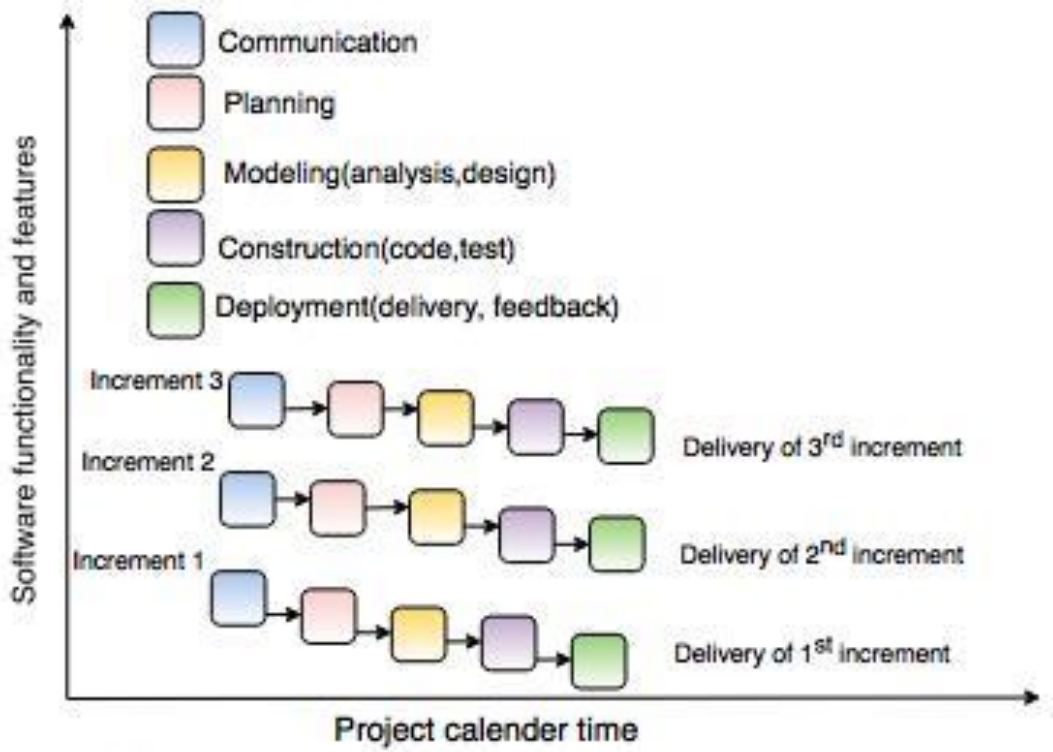


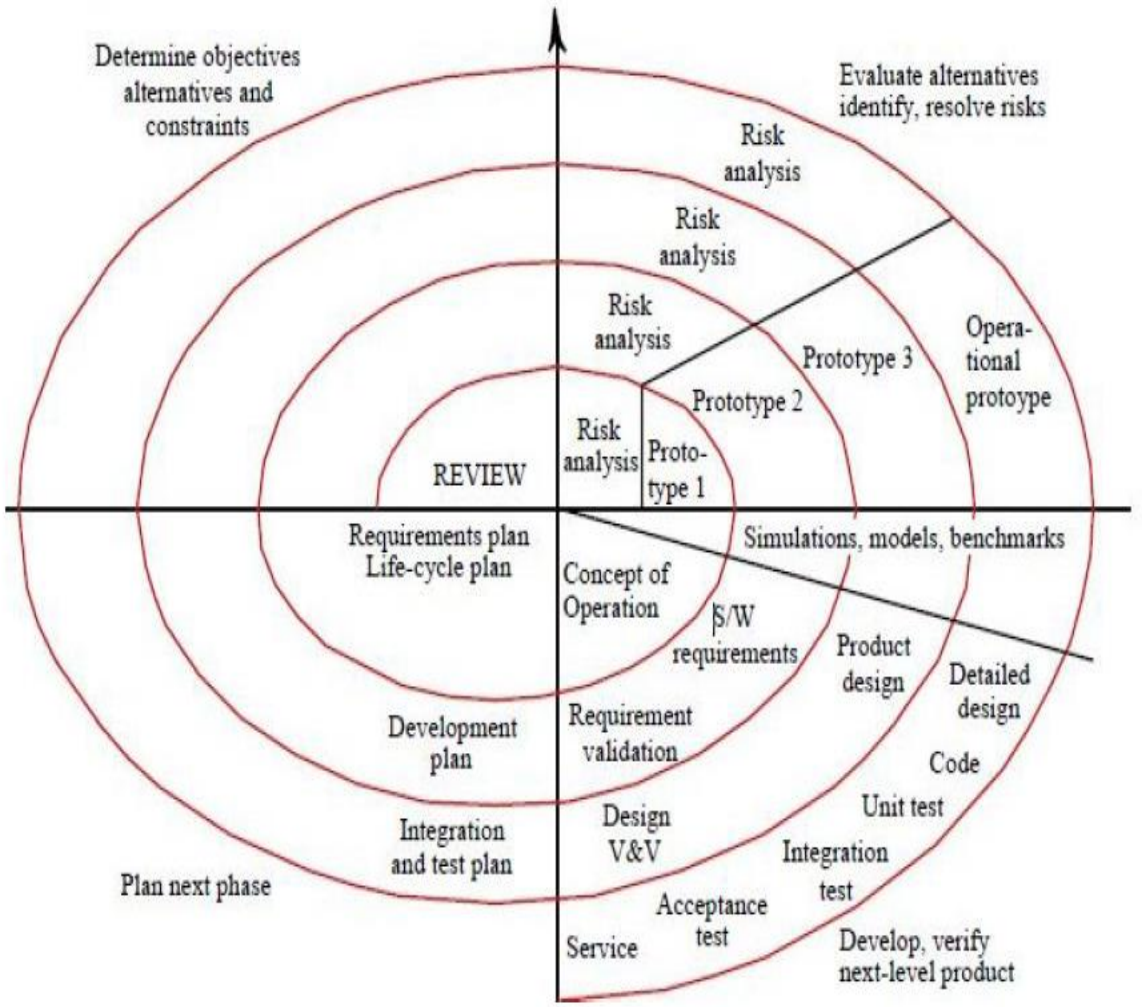
Fig. - Incremental Process Model

Spiral development

- The spiral model is a risk-driven where the process is represented as spiral rather than a sequence of activities.
- The spiral model was originally proposed by Boehm in 1988.
- In this model, process is represented as a spiral rather than as a sequence of activities with backtracking from one activity to another.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required
- Risks are explicitly assessed and resolved throughout the process
- It was designed to include the best features from the waterfall and prototyping models, and introduces a new component; risk-assessment.

Structure/phases

- **Determine Objectives, alternatives and Constraints: (Specific objectives for the phase are identified)**
  - Objectives: functionality, performance, hardware/software interface, critical success factors, etc.
  - Alternatives: build, reuse, buy, sub-contract, etc.
  - Constraints: cost, schedule, interface, etc.
- **Evaluate alternatives ,identify, resolve risks: Risks are assessed and activities put in place to reduce the key risks)**
  - Study alternatives relative to objectives and constraints
  - Identify risks (lack of experience, new technology, tight schedules, poor process, etc.
  - Resolve risks (evaluate if money could be lost by continuing system development
- **Development and validation: A development model for the system is chosen which can be any of the generic models**
  - After risk evaluation, a development model for the system is chosen which can be any generic model. After risk evaluation, a **process model for the system is chosen**. So if the risk is expected in the user interface then we must **prototype** the user interface. If the risk is in the development process itself then use the **waterfall model**.
- **Planning for next step: (The project is reviewed and the next phase of the spiral is planned)**
  - The project is reviewed and a decision made whether to continue with a further loop of the spiral. If it is decided to continue, plans are drawn up for the next phase of the project.
  - Develop various types of plan like project plan, configuration management plan,test plan, installation plan etc.



Spiral model

Advantages

- The model tries to resolve all possible risks involved in the project starting with the highest risk.
- Provides early indication of insurmountable risks, without much cost
- Users see the system early because of rapid prototyping tools
- Critical high-risk functions are developed first
- The design does not have to be perfect
- Users can be closely tied to all lifecycle steps
- Early and frequent feedback from users
- Cumulative costs assessed frequently

Disadvantages

- The model requires expertise in risk management and excellent management skills.
- Time spent for evaluating risks too large for small or low-risk projects
- Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive
- The model is complex
- Spiral may continue indefinitely
- Developers must be reassigned during non-development phase activities
- May be hard to define objective, verifiable milestones that indicate readiness to proceed through the next iteration

When to use Spiral model?

- When creation of a prototype is appropriate
- For medium to high-risk projects
- Requirements are complex
- When costs and risk evaluation is important
- Users are unsure of their needs
- New product line

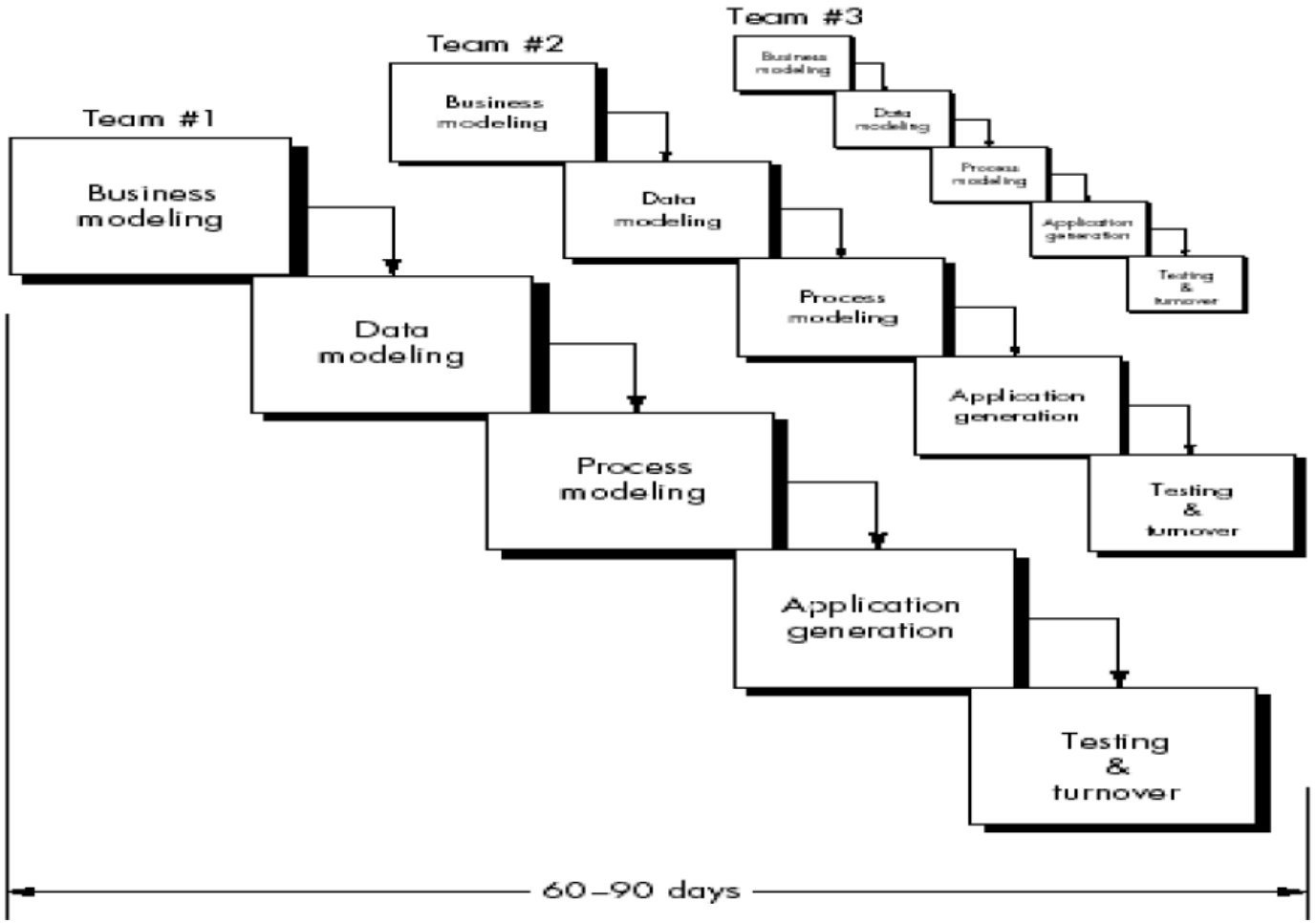
Significant changes are expected(Research and Exploration)

- Long-term project commitment unwise because of potential changes to economic priorities

Rapid Application Development Model

- The Rapid Application Development (or RAD) model is based on prototyping and iterative model with no (or less) specific planning.
- In RAD model, there is less attention paid to the planning and more priority is given to the development tasks.
- It targets at developing software in a short span of time.
- RAD allows project managers and stakeholders to accurately measure progress and communicate in real time on evolving issues or changes.
- The RAD model is a “high-speed” adaptation of the linear sequential model in which rapid development is achieved by using component-based construction.
- If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a “fully functional system” within very short time periods (e.g., 60 to 90 days)
- RAD (Rapid Application Development) 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

RAD MODEL


Team #1
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
Team #2
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
Team #3
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
60–90 days

Phases:

- **Business modeling:**
  - The information flow among business functions is modeled in a way that answers the following questions:
    - What information drives the business process?
    - What information is generated? Who generates it?
    - Where does the information go? Who processes it?
- **Data modeling:**
  - The information flow defined as part of the business modeling phase is refined into a set of data objects that are needed to support the business.
  - The characteristics (called *attributes) of each object are identified and the relationships between these objects defined.*
- **Process modeling:**
  - The data objects defined in the data modeling phase are transformed to achieve the information flow necessary to implement a business function.
  - Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.
- **Application generation:**
  - RAD assumes the use of fourth generation techniques).
  - Rather than creating software using conventional third generation programming languages the RAD process works to reuse existing program components (when possible) or create reusable components (when necessary).
  - In all cases, automated tools are used to facilitate construction of the software.
- **Testing and turnover:**
  - Since the RAD process emphasizes reuse, many of the program components have already been tested. This reduces overall testing time.
  - However, new components must be tested and all interfaces must be fully exercised. Obviously, the time constraints imposed on a RAD project demand “scalable scope”. If a business application can be modularized in a way that enables each major function to be completed in less than three months, it is a candidate for RAD. Each major function can be addressed by a separate RAD team and then integrated to form a whole.

- When to use RAD Methodology?
  - When a system needs to be produced in a short span of time (2-3 months)
  - When the requirements are known
  - When the user will be involved all through the life cycle
  - When technical risk is less
  - When a budget is high enough to afford designers for modeling along with the cost of automated tools for code generation
- Advantages
  - Flexible and adaptable to changes
  - Makes use of reusable components, to decrease the cycle time
  - It is useful when you have to reduce the overall project risk
  - Due to code generators and code reuse, there is a reduction of manual coding
- Disadvantages
  - Not Suitable for smaller Projects
  - For large but scalable projects, RAD requires sufficient human resources to create the right number of RAD teams.
  - If commitment is lacking from customers or developers, RAD projects will fail.
  - RAD is not appropriate when technical risks are high. Heavy use of New technology requires high degree of interoperability.

Agile Methodology

- AGILE methodology is a practice that promotes **continuous iteration** of development and testing throughout the software development lifecycle of the project.
- Agile methods **break tasks into smaller iterations**, or parts do not directly involve **long term planning**. The **project scope and requirements** are laid down **at the beginning** of the development process. **Plans** regarding the number of iterations, the duration and the scope of each iteration are **clearly defined in advance**.
- Each iteration is considered as a **short time "frame**" in the Agile process model, which typically lasts from one to four 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.
- In the Agile model, both development and testing activities are concurrent, unlike the Waterfall model.
- **Scrum and Kanban** are two of the most widely used Agile methodologies.
- The agile software development emphasizes on four core values.
  - Individual and team interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
- Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

Graphical Illustration of Agile model

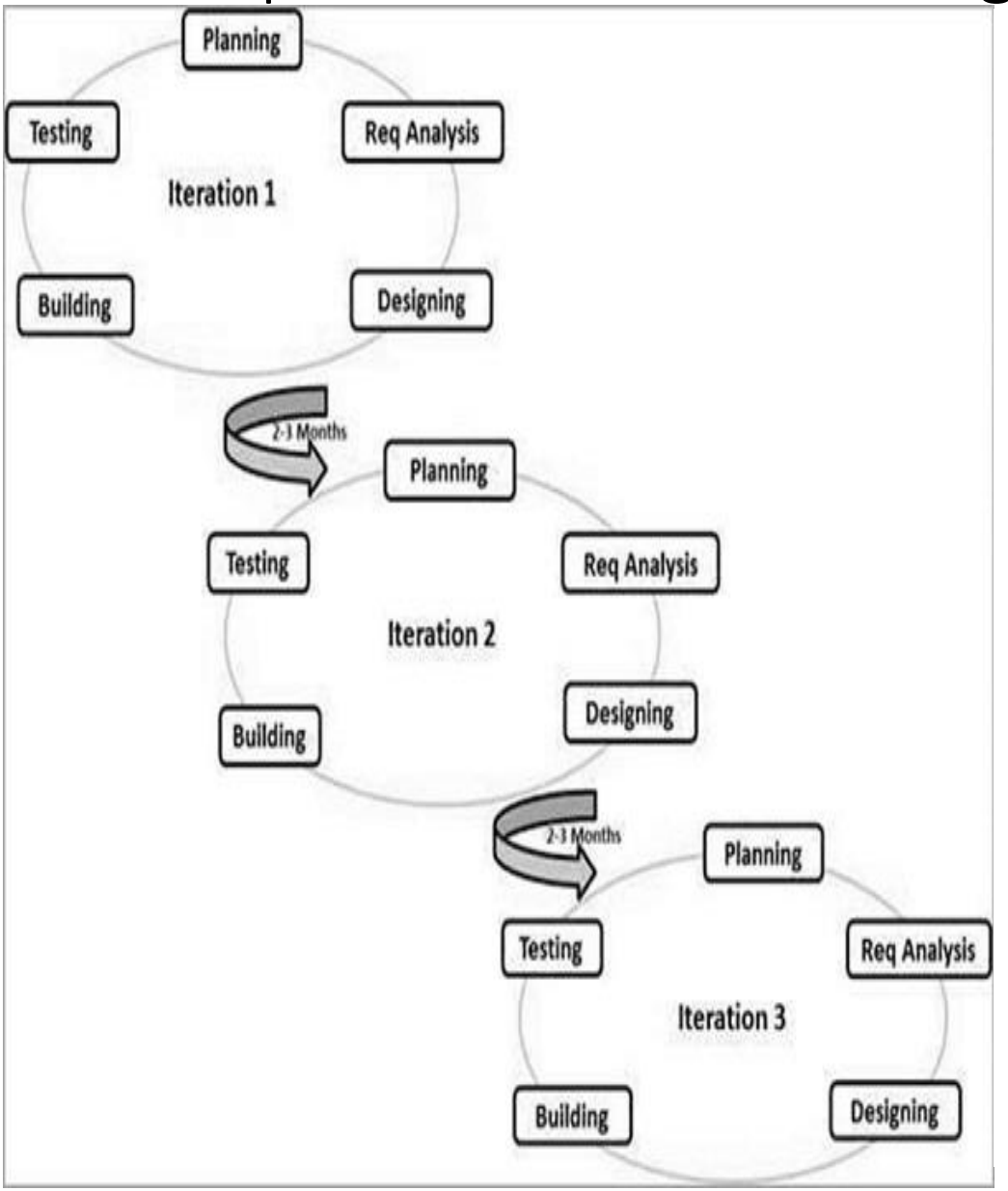


Fig. Agile Model

Phases of Agile model

- **Requirements gathering:**
  - In this phase, you must define the requirements. You should explain **business opportunities and plan the time and effort** needed to build the project. Based on this information, you can evaluate technical and economic feasibility.
- **2. Design the requirements:**
  - When you have identified the project, work with stakeholders to define requirements. You can use the **user flow diagram or the high-level UML diagram** to show the work of new features and show how it will apply to your existing system.
- **3. Construction/ iteration:**
  - When the team defines the requirements, the work begins. **Designers and developers start working on their project**, which aims to deploy a working product. The product will undergo **various stages of improvement**, so it includes simple, minimal functionality.
- **4. Testing:**
  - In this phase, the Quality Assurance team examines the product's performance and looks for the bug.
- **5. Deployment:**
  - In this phase, the team issues a product for the user's work environment.
- **6. Feedback:**
  - After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

- The most popular Agile methods include
  - Rational Unified Process(RUP) (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995).
- Agile Manifesto:
  - **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.

- **Agile Principles:**
  1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
  2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
  3. Deliver working software frequently, from a couple of weeks to a couple of months, with preference to the shorter timescale.
  4. Business people and developers must work together daily throughout the project.
  5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
  6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
  7. Working software is the primary measure of progress.
  8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
  9. Continuous attention to technical excellence and good design enhances agility.
  10. Simplicity -- the art of maximizing the amount of work not done -- is essential.
  11. The best architectures, requirements, and designs emerge from self-organizing teams.
  12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

- **When to use agile?**
  - 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.

Benefits of Agile

CUSTOMER

- More responsive to requests
- High-value features
- Delivered more quickly with short cycles

DEVELOPMENT TEAMS

- Enjoy development work
- Work is valued and used
- Reduced non-productive work

SCRUMMASTER

- Planning/task-level tracking in daily meetings
- Tremendous awareness of project state/status
- Catching and addressing issues quickly

VENDOR

- Focused development on high-value features
- Increased efficiency
- Reduce wastage and decreased overhead

PRODUCT OWNER

- Development work aligns with customer needs
- Frequent opportunities to re-prioritize work
- Maximum delivery of value

PMOS AND C-LEVEL EXECUTIVES

- High visibility of daily project development
- Adjust strategies based on hard information
- Plan effectively with less speculation

Agile methodology

Advantages

- Is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- 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.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers thus reducing development time.
- Frequent Delivery
- Face-to-Face Communication with clients.
- Efficient design and fulfils the business requirement.
- Anytime changes are acceptable.

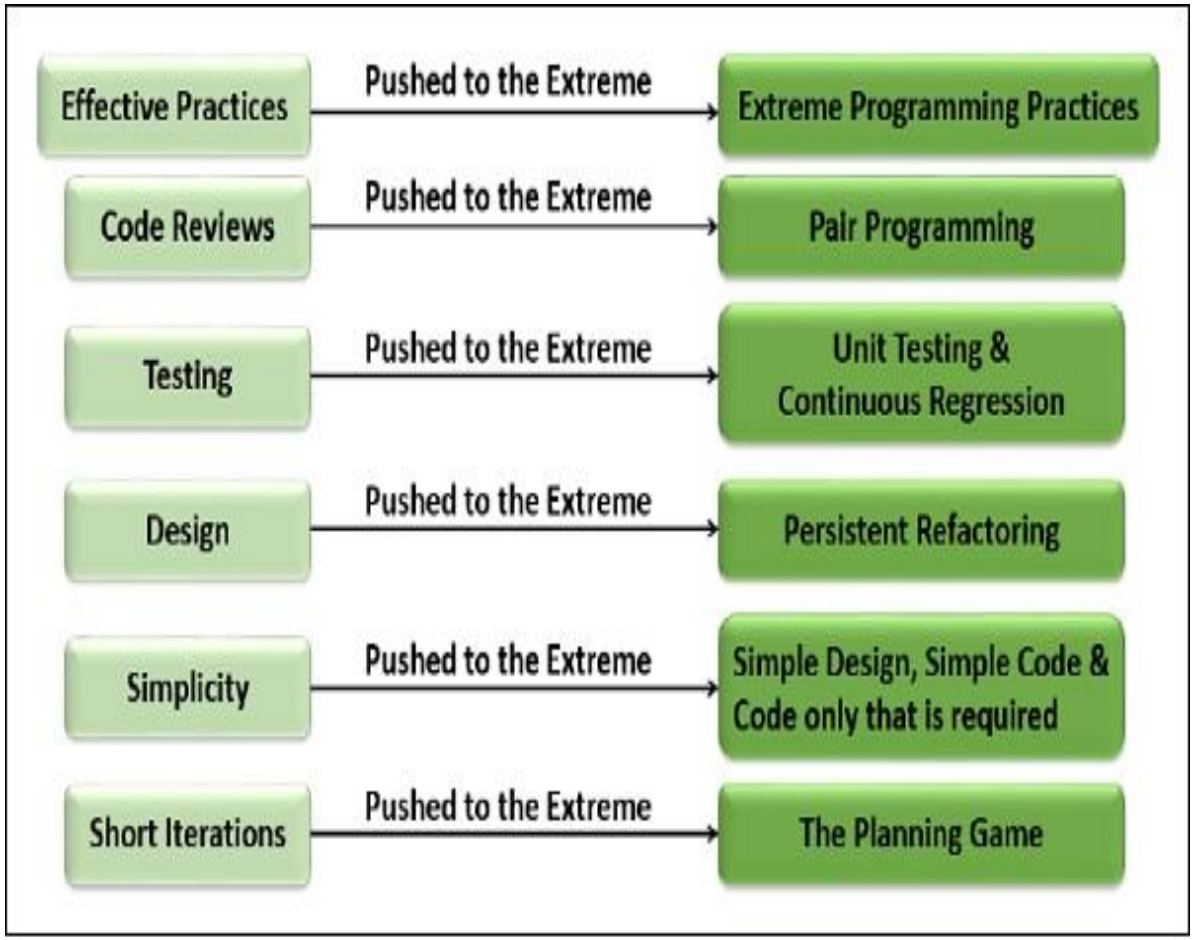
Disadvantages

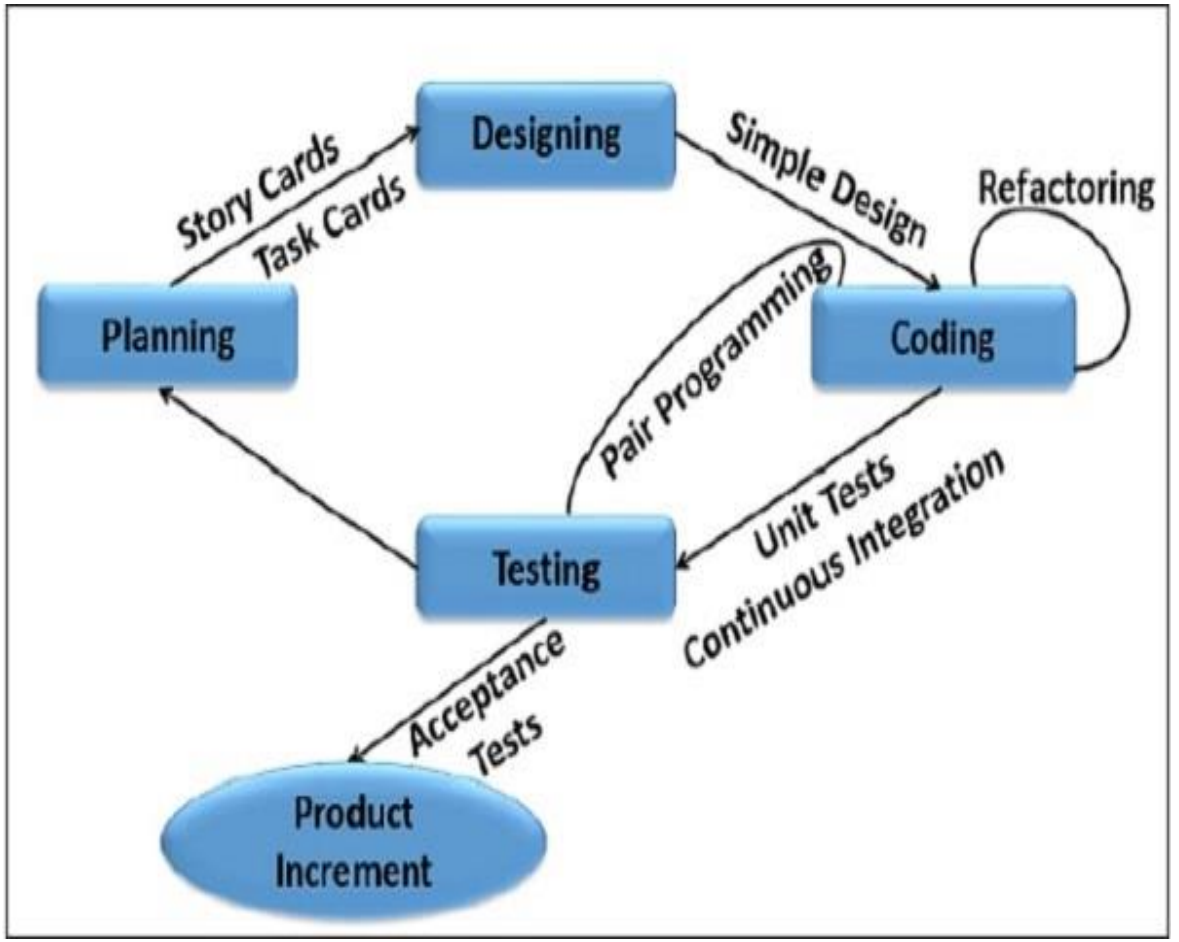
- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.
- 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.

Extreme Programming(XP)

- Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, and higher quality of life for the development team. XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development.
- XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop a software
- **When XP is applicable?**
  - Dynamically changing software requirements
  - Risks caused by fixed time projects using new technology
  - Small, co-located extended development team
  - The technology you are using allows for automated unit and functional tests
- **What are the values of XP?**
  - Communication
    - face to face discussion with the aid of a white board or other drawing mechanism;
    - must be in communication with each other at all times
  - Simplicity
    - address only the requirements that you know about; don’t try to predict the future
  - Feedback
    - builds something, gathers feedback on your design and implementation, and then adjust your product going forward.
  - Courage
    - effective action in the face of fear
  - Respect
    - respect each other in order to communicate with each other, provide and accept feedback that honors your relationship, and to work together to identify simple designs and solutions.

Why Extreme?


Effective Practices
Pushed to the Extreme
Extreme Programming Practices
Code Reviews
Pushed to the Extreme
Pair Programming
Testing
Pushed to the Extreme
Unit Testing & Continuous Regression
Design
Pushed to the Extreme
Persistent Refactoring
Simplicity
Pushed to the Extreme
Simple Design, Simple Code & Code only that is required
Short Iterations
Pushed to the Extreme
The Planning Game


Designing
Story Cards
Task Cards
Simple Design
Refactoring
Planning
Pair Programming
Coding
Testing
Unit Tests
Continuous Integration
Acceptance Tests
Product Increment

Advantages

- Lightweight methods suit small-medium size projects.
- Produces good team cohesion.
- Emphasizes final product.
- Iterative.
- Test based approach to requirements and quality assurance.
- Close contact with the customer
- No unnecessary programming work
- Stable software through continuous testing
- Error avoidance through pair programming
- Changes can be made at short notice

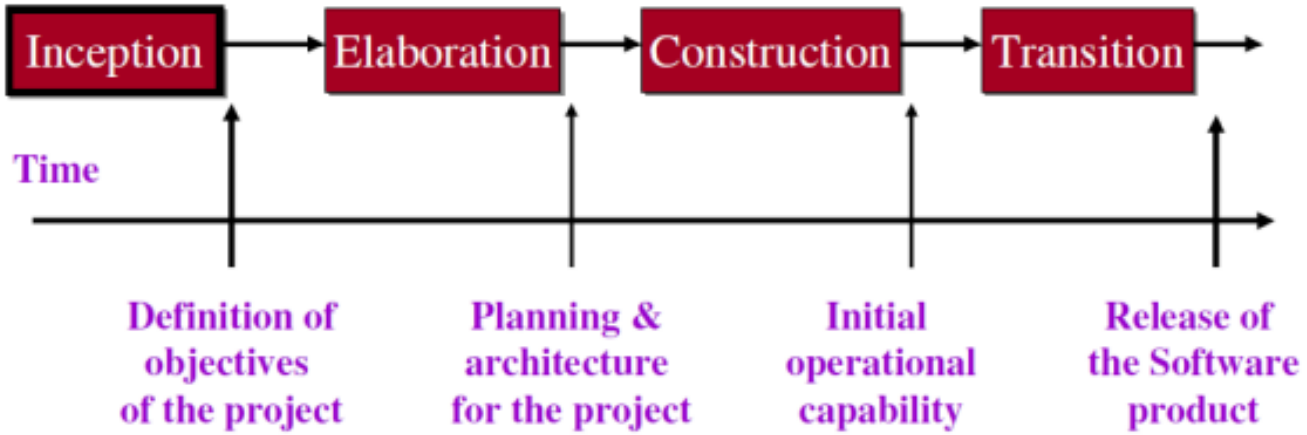
Disadvantages

- Difficult to scale up to large projects where documentation is essential.
- Needs experience and skill if not to degenerate into code-and-fix.

Rational Unified Process(RUP)

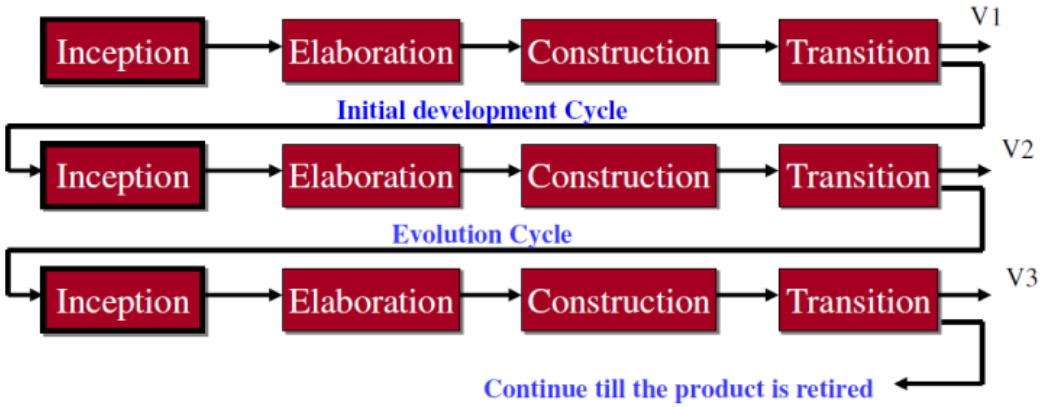
- **Rational Unified Process (RUP)** is a software development process for object-oriented models. It is also known as the **Unified Process Model**.
- Developed by I.Jacobson, G.Booch and J.Rumbaugh.
- Software engineering process with the goal of producing good quality maintainable software within specified time and budget.
- Developed through a series of fixed length mini projects called iterations.
- Maintained and enhanced by Rational Software Corporation and thus referred to as **Rational Unified Process (RUP).**
- The Rational Unified Process (RUP) is **iterative**, meaning repeating; and **agile**. Iterative because all of the process's core activities repeat throughout the project. The process is agile because various components can be adjusted, and phases of the cycle can be repeated until the software meets requirements and objectives.
- RUP reduces unexpected development costs and prevents wastage of resources.
- RUP has the following key characteristics:
  - Use-case driven from inception to deployment
  - Architecture-centric, where architecture is a function of user needs
  - Iterative and incremental, where large projects are divided into smaller projects

Phases of RUP:



- **Inception: (The core idea is envisioned)**
  - defines scope of the project.
- **Elaboration: (Use cases and Architecture are designed)**
  - How do we plan & design the project?
  - What resources are required?
  - What type of architecture may be suitable?
- **Construction: (Activities from design to completed products)**
  - the objectives are translated in design & architecture documents.
- **Transition : (Follow up activities to ensure customer satisfaction)**
  - involves many activities like delivering, training, supporting, and maintaining the product.

Fig: Initial Development and Evolution Cycle of RUP



V1=version1. V2 =version2. V3=version3

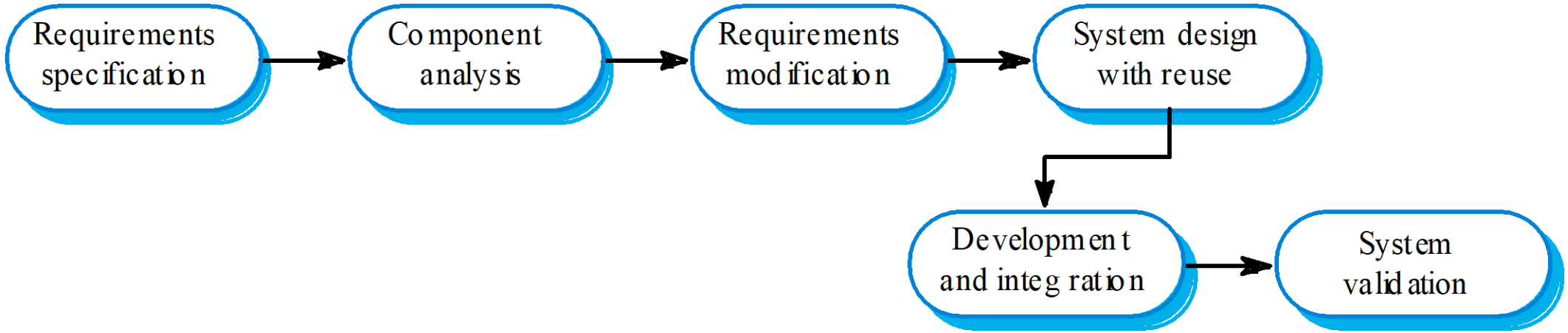
- **Advantages**
  - It allows you to deal with changing requirements regardless of whether they are coming from the customer or from the project itself.
  - It emphasizes the need for accurate documentation.
  - It forces integration to happen throughout the software development, more specifically in the construction phase.
- **Disadvantages**
  - It mostly relies on the ability of experts and professionals to assign the activities to individuals who should then produce pre-planned results in the form of artifacts.
  - The integration in development process can also have an adverse impact on some more fundamental activities during the stages of testing

Component Based Software Engineering(CBSE)

- **Component Based Software Engineering (CBSE)** is a process that focuses on the design and development of computer-based systems with the use of reusable software components.
- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Component-based software engineering (CBSE) is an approach to software development that relies on software reuse.
- It emerged from the failure of object-oriented development to support effective reuse. Single object classes are too detailed and specific.
- Components are more abstract than object classes and can be considered to be stand-alone service providers.

CBSE Process

- When reusing components, it is essential to make trade-offs between ideal requirements and the services actually provided by available components.
- This involves following process stages:
  - Developing outline requirements:**component analysis**
  - Searching for components then modifying requirements according to available functionality:**requirement modifications**
  - Searching again to find if there are better components that meet the revised requirements:**system design with reuse**
  - Develop all the components and integrate them to produce a final model:**Development and Integration**
  - Check the model based on system requirements and users requirement prespectives:**system validation**


Requirements specification
Component analysis
Requirements modification
System design with reuse
Development and integration
System validation

When to use CBSE?

-The working environment where the concept of reusability can be implemented
-Special skilled people available
-High quality product requirements

- **Benefits of CBSE**
  - Increase Reusability
  - Reduction of Development time
  - Cost reduction
  - Quality Improvements
- **Challenges of CBSE**
  - Difficulty in identification of proper components
  - Requirements ambiguity
  - Issues in components interoperability
  - Lack of detailing of the components makes it difficult to fix problems or issues

CASE and CASE Tools

- **Computer-aided software engineering (CASE)**
  - Facilitate creation of a central repository for system descriptions and specifications
  - Computer-aided software engineering (CASE) is the scientific application of a set of tools and methods to a software system which is mean to result in high-quality, defect-free, and maintainable software products.
- **CASE Tools and its classification**
  - Software tools providing automated support for systems development
  - The CASE Tools are capable of automating the tasks in almost every phase of software development life cycle reducing the workload of developers and other team members. They are application programs which are designed to perform specific tasks.
  - By automating the various processes of software development life cycle, the risk of errors & fault decreases and efficiency & functionality increases. Also, CASE Tools are efficient enough to reduce the time consumption effectively making the cost of the project in control
  - Project dictionary/workbook: system description and specifications
  - **Upper CASE Tools**: Planning, analysis, and designing of different stages of the software development life cycle can be performed using upper case tools.
  - **Lower Case Tools**: Implementation, testing, and maintenance can be performed using lower case.
  - **Diagramming tools**
  - Example :Oracle Designer, Rational Rose

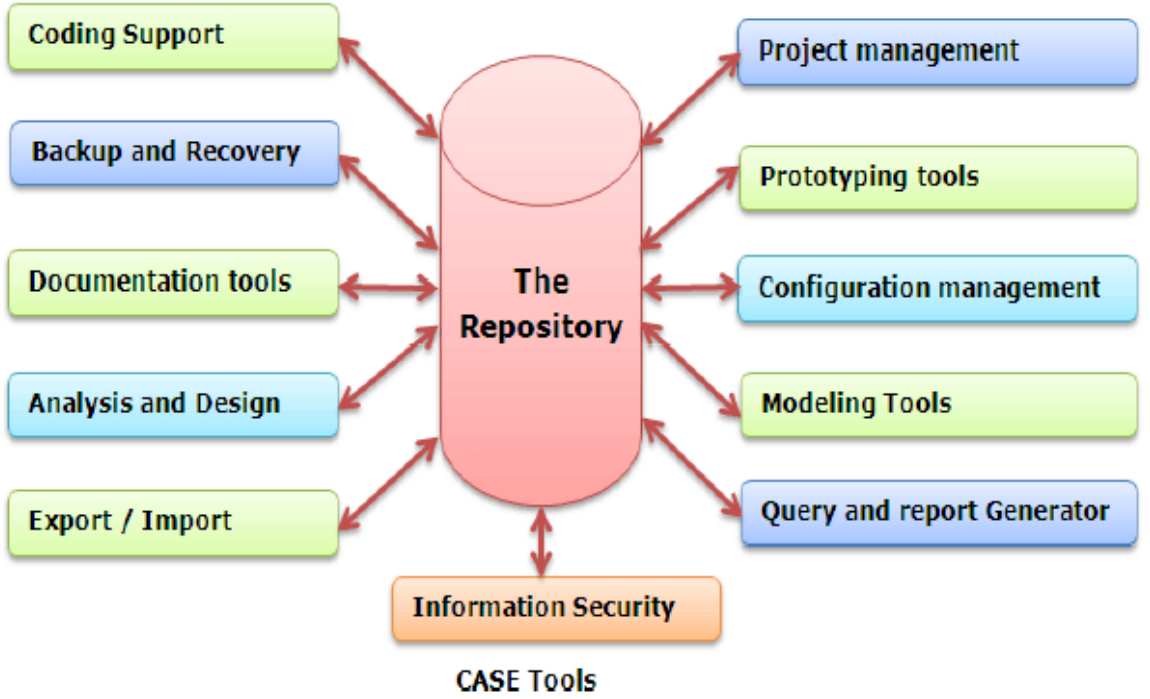
  Different IDEs like Eclipse, configuration tools like Git ,word processor, excel sheet etc.

Today's CASE tools provide two distinct ways to develop system models **– forward engineering and reverse engineering**.

- **Forward engineering** requires the system analyst to draw system models, either from scratch or from templates. The resulting models are subsequently transformed into program code.
- **Reverse engineering**, on the other hand, allows a CASE tool to read existing program code and transform that code into a representative system model that can be edited and refined by the systems analyst.
- CASE tools that allow for bi-directional, forward and reverse engineering are said to provide for

  "Round -trip Engineering."

- **Components of CASE:**
- At the center of any CASE tool's architecture is a developer's database called a **CASE repository**.
- **CASE repository** is a system developer's database where developers can store system models, detailed description and specification, and other products of system development.
- It is also called dictionary or encyclopedia. Around the CASE repository is a collection of tools or facilities for creating system models and documentation.
- These facilities generally include:
  - **Diagramming tools** –These tools are used to draw system models. Eg visio
  - **Dictionary tools** – These tools are used to record, delete, edit, and output detailed documentation and specification
  - **Design tools** – These tools are used to construct system components including system inputs and outputs. These are also called prototyping tools. Eg Photoshop
  - **Documentation tools** – These tools are used to assemble, organize, and report on system models, descriptions and specifications, and prototypes. Eg webhelp
  - **Quality management tools** – These tools are used to analyze system models, descriptions and specifications, and prototypes for completeness, consistency, and conformance to accepted rules of methodologies. Eg Selenium
  - **Design and code generator tools** – These tools automatically generate database designs and application programs or significant portions of those programs.



CASE Tools

- **Analysis CASE Tools** : These tools are used to check the software product for its proper functionality and requirement analysis For example : Artemis
- **Project Management Tools :** These CASE tools are used in planning the whole software product with respect to its features, functionalities, available resources, budget, cost, time etc.for example :Dashboard,Project Scheduling etc
- **Prototyping Tools :** These CASE tools are used to generate prototypes for various software modules of any software product. For example : Framer, Principle etc

- **Advantages of CASE**
  - As special emphasis is placed on redesign as well as testing, the servicing cost of a product over its expected lifetime is considerably reduced.
  - The overall quality of the product is improved as an organized approach is undertaken during the process of development.
  - Chances to meet real-world requirements are more likely and easier with a computer-aided software engineering approach.
- **Weaknesses**
  - Costly
  - Tools availability and utilization