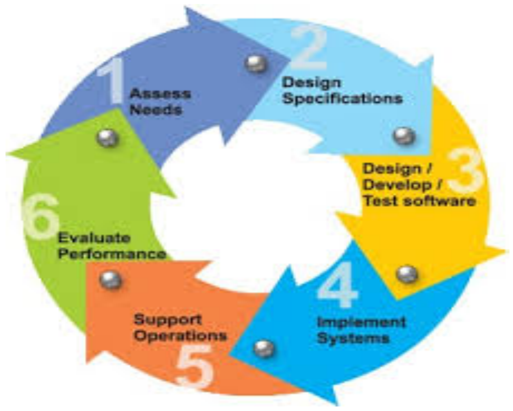
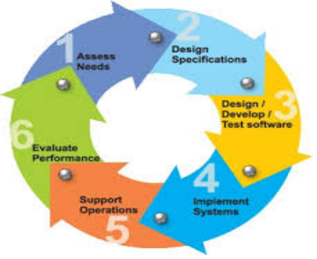


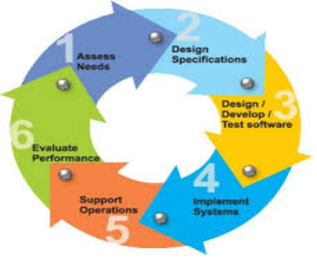
Week-2: Lesson-1 Process Models

Jawad Ibn Ahad
Lecturer
Department of Computer Science and Engineering
Daffodil International University
Email: jahad.cse0544.c@diu.edu.bd



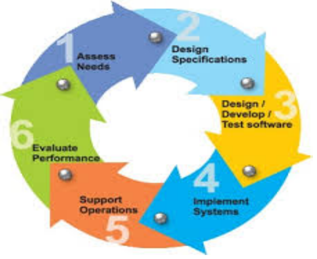
Topics Covered

- Software Process
- Process Model
- A generic Process Model
- Software Framework Activities
- Software Process Model
- Selection of Process Model



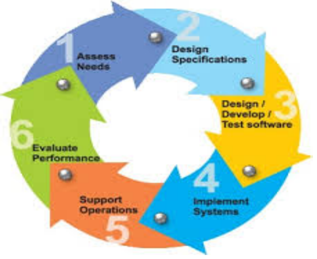
Definition of Software Process

- ***Software Process:***
  - A **framework** for the activities, actions, and tasks that are required to build high-quality software.
  - SP defines the approach that is taken as software is engineered.
  - Is not equal to software engineering, which also encompasses **technologies** that populate the process– technical methods and automated tools.
- ***Process Model :***
  - A Process Model describes the **sequence of phases** for the entire lifetime of a product. Therefore it is sometimes also called Product Life Cycle.
  - This covers everything from the initial commercial idea until the final de-installation or disassembling of the product after its use.



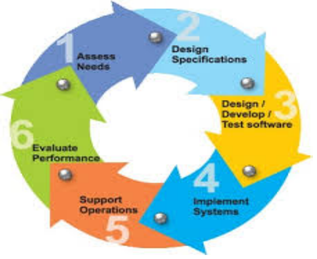
What / who / why is Process Models?

- ***What***: Go through a series of predictable steps--- a road map that helps you create a timely, high-quality results.
- ***Who***: Software engineers and their managers, clients also. People adapt the process to their needs and follow it.
- ***Why***: Provides stability, control, and organization to an activity that can if left uncontrolled, become quite chaotic. However, modern software engineering approaches must be agile and demand ONLY those activities, controls and work products that are appropriate.
- ***What Work products***: Programs, documents, and data
- ***What are the steps:*** The process you adopt depends on the software that you are building. One process might be good for aircraft avionic system, while an entirely different process would be used for website creation.
- ***How to ensure right***: A number of software process assessment mechanisms that enable us to determine the maturity of the software process. However, the quality, timeliness and long-term viability of the software are the best indicators of the efficacy of the process you use.



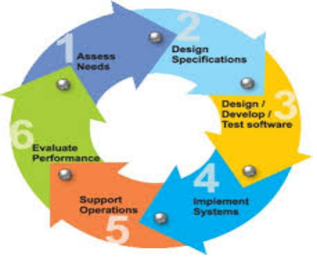
A Generic Process Model

- As we discussed before, a generic process framework for software engineering defines five framework activities-communication, planning, modeling, construction, and deployment.
- In addition, a set of umbrella activities- project tracking and control, risk management, quality assurance, configuration management, technical reviews, and others are applied throughout the process.

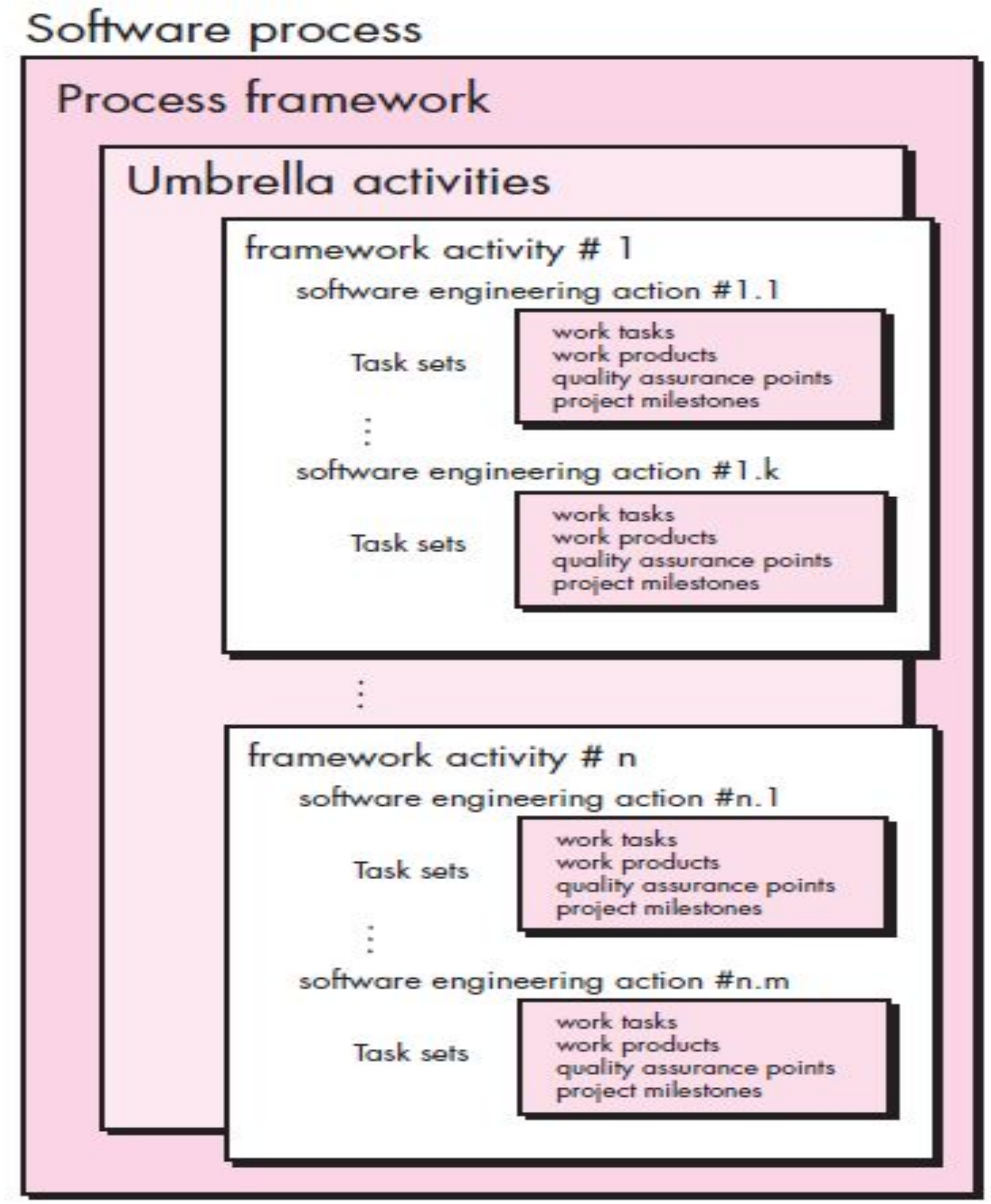


A Generic Process Model

- ***Communication :*** This activity involves heavy communication with customers and other stakeholders in order to gather requirements and other related activities.
- ***Planning :*** Here a plan to be followed will be created which will describe the technical tasks to be conducted, risks, required resources, work schedule etc.
- ***Modeling :*** A model will be created to better understand the requirements and design to achieve these requirements.
- ***Construction :*** Here the code will be generated and tested.
- ***Deployment :*** Here, a complete or partially complete version of the software is represented to the customers to evaluate and they give feedbacks based on the evaluation.

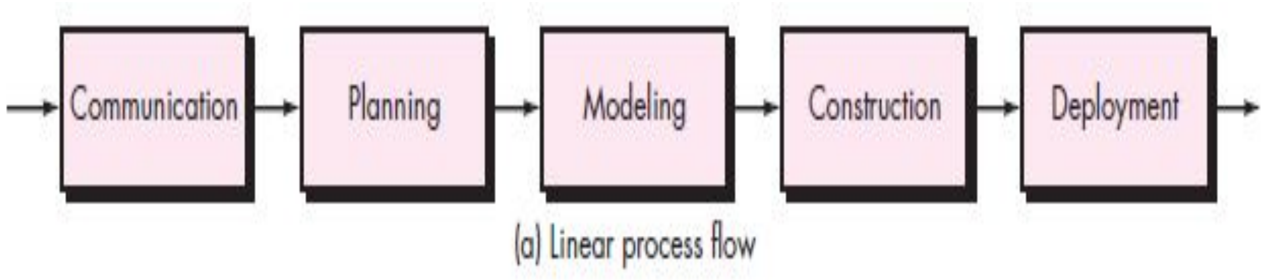


A Generic Process Model

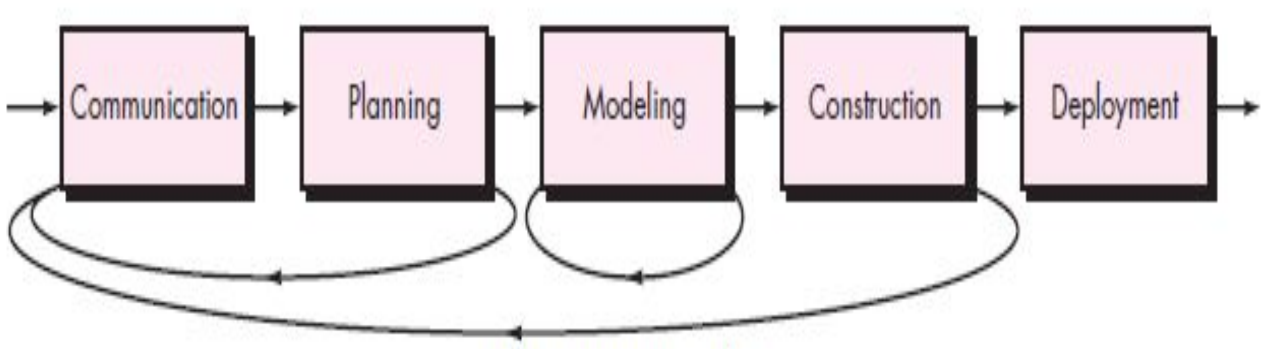


Framework Activities

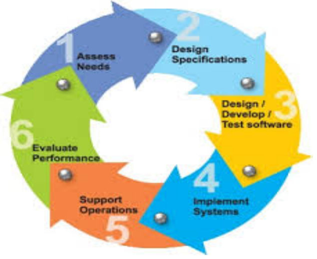
Process Flow



(a) Linear process flow

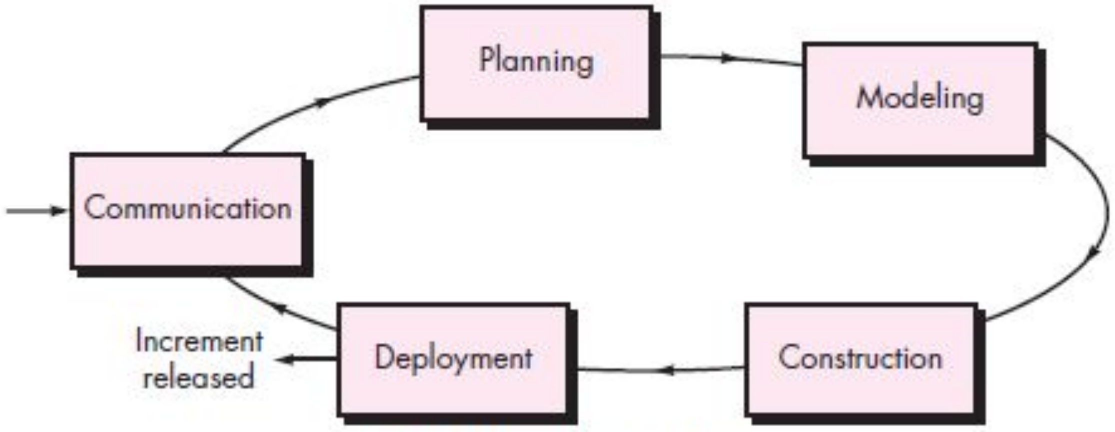


(b) Iterative process flow

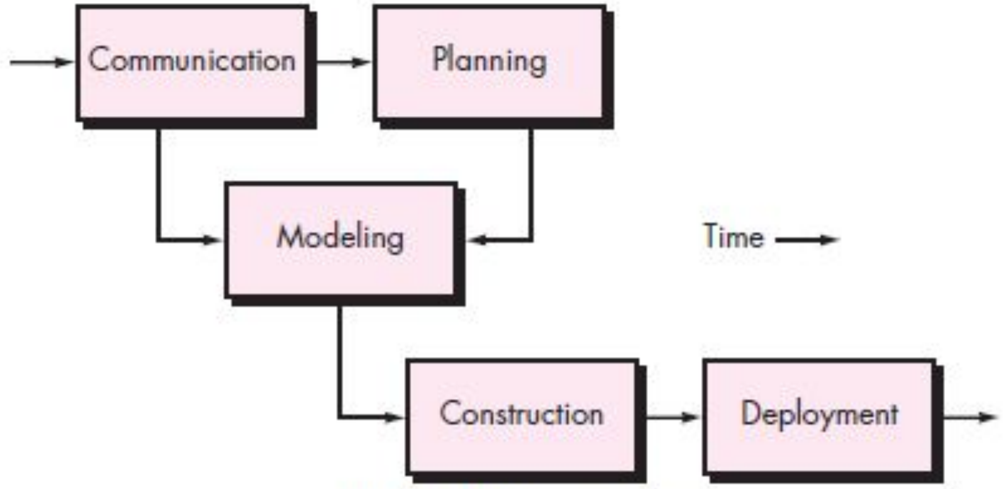


Process Flow

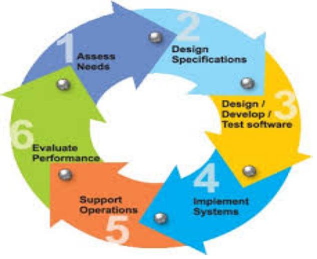
Process Flow



(c) Evolutionary process flow

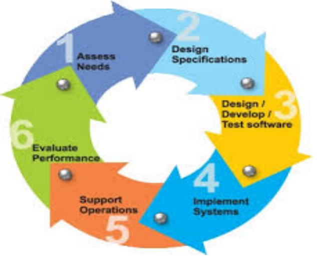


(d) Parallel process flow



Process Flow

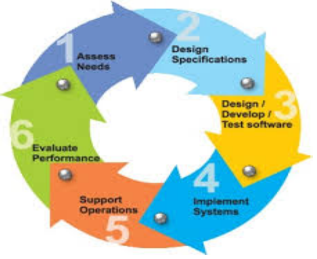
- Linear process flow executes each of the five activities in sequence.
- An iterative process flow repeats one or more of the activities before proceeding to the next.
- An evolutionary process flow executes the activities in a circular manner. Each circuit leads to a more complete version of the software.
- A parallel process flow executes one or more activities in parallel with other activities (modeling for one aspect of the software in parallel with construction of another aspect of the software.



Identifying a Task Set for Small Project

- For example, a small software project requested by one person with simple requirements, the communication activity might encompass little more than a phone all with the stakeholder. Therefore, the only necessary action is phone conversation, the work tasks of this action are:

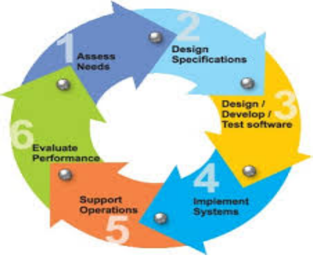
1. ***Make contact with stakeholder via telephone.***
2. ***Discuss requirements and take notes.***
3. ***Organize notes into a brief written statement of requirements.***
4. ***E-mail to stakeholder for review and approval.***



Example of a Task Set for Mid Level Project

- The task sets for Requirements gathering action for a **simple** project may include:

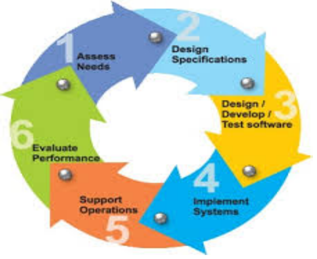
1. ***Make a list of stakeholders for the project.***
2. ***Invite all stakeholders to an informal meeting.***
3. ***Ask each stakeholder to make a list of features and functions required.***
4. ***Discuss requirements and build a final list.***
5. ***Prioritize requirements.***
6. ***Note areas of uncertainty***.



Example of a Task Set for a Big Project

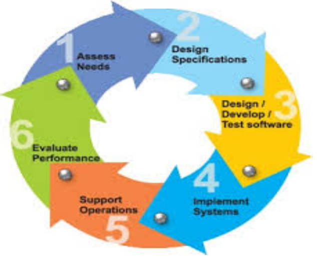
- **The task sets for Requirements gathering action for a big project may include:**

1. ***Make a list of stakeholders for the project.***
2. ***Interview each stakeholders separately to determine overall wants and needs.***
3. ***Build a preliminary list of functions and features based on stakeholder input.***
4. ***Schedule a series of facilitated application specification meetings.***
5. ***Conduct meetings.***
6. ***Produce informal user scenarios as part of each meeting.***
7. ***Refine user scenarios based on stakeholder feedback.***
8. ***Build a revised list of stakeholder requirements.***
9. ***Use quality function deployment techniques to prioritize requirements.***
10. ***Package requirements so that they can be delivered incrementally.***
11. ***Note constraints and restrictions that will be placed on the system.***
12. ***Discuss methods for validating the system.***



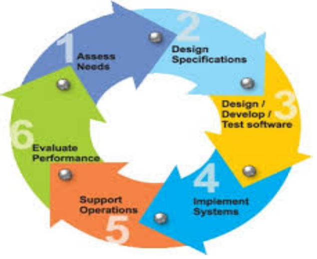
Process Patterns

- A *process pattern:*
  - A **pattern** is a generalized set of predefined steps, that solve a specific problem or achieve a goal, in an organized fashion.
  - You can think of patterns like cake recipes. If you follow them to the letter, you will get the exact same result you expect, a mouth-watering cake.
  - Process patterns are similar, but they deal with a more specific topic.
  - A **process pattern** is a group of proven steps, that complete a specific task or tasks, and provide a consistently favorable result for a common problem.
  - In other words, they are a template for achieving your specific goal consistently.



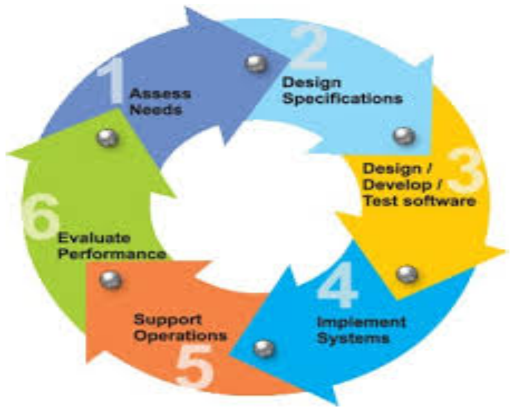
Process Patterns

- Template for describing a process pattern:
  - ***Pattern Name:***
    - The pattern name is given a meaningful name that describe its function within software process.
    - Example: Customer-Communication
  - ***Intent:***
    - The objective of this pattern is describe briefly.
    - Example: Intent of the communication customer to connection establish with customer.
  - ***Type:***
    - The pattern type is specified.
    - Three types:
      - **Task Pattern** - action or work task
      - **Stage Pattern** – frame activity.
      - **Phase Pattern** – sequence of frame.



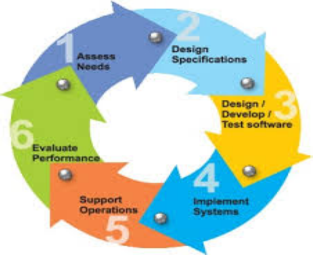
Process Patterns (Cont…)

- ***Initial Context:***
  - Describe which pattern are applies
  - Prior to the initialization of the pattern.
- ***Resulting Context:***
  - The condition that will result once the pattern has been successfully implementers are describe.
- ***Related Pattern:***
  - A list of process pattern that are directly related to this one are provided as a hierarchy on in some other diagrammatic form.



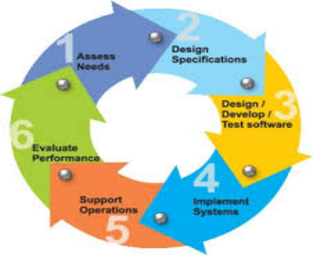
Week-2: Lesson-2
Process Models

Jawad Ibn Ahad
Lecturer
Department of Computer Science and Engineering
Daffodil International University
Email: jahad.cse0544.c@diu.edu.bd

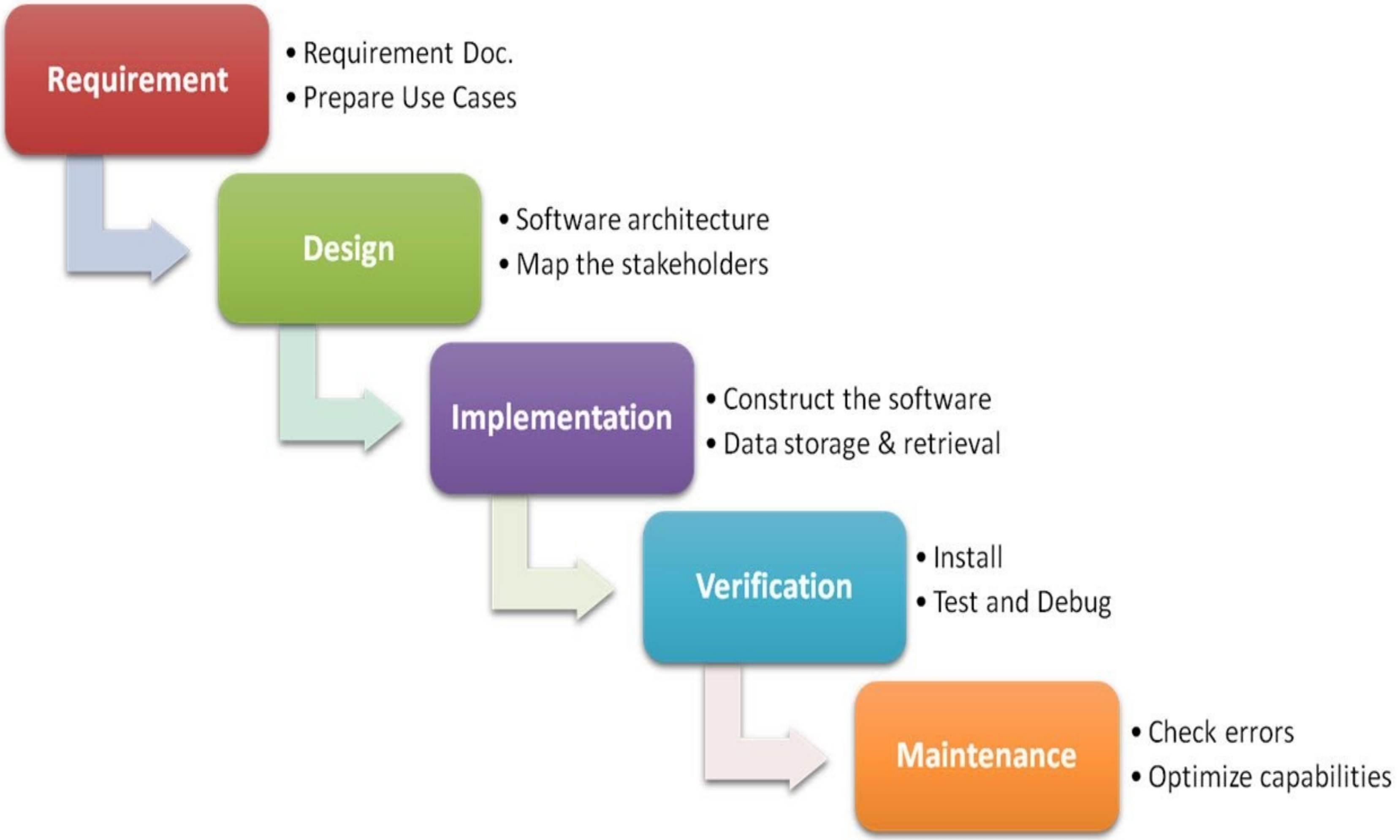


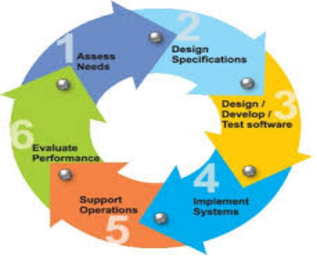
Software Process Model

- A Process Model describes the sequence of phases for the entire lifetime of a product.
- It is sometimes also called Product Life Cycle.
- This covers everything from the initial commercial idea until the final de-installation or disassembling of the product after its use
- All software process models can accommodate the generic framework activities.



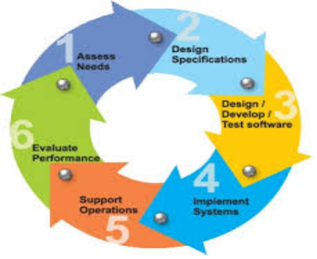
Waterfall Model





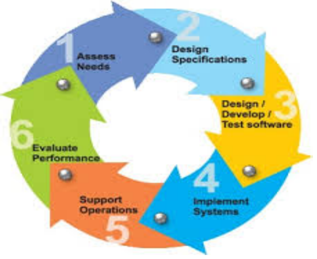
Waterfall Model (Cont..)

- The *first published model* of the software development process was derived from more general system engineering processes (Royce, 1970).
- This model is known as the '*waterfall model*' or software life cycle
- *The waterfall model* This takes the fundamental process activities of specification, development, validation, and evolution and represents them as separate process phases such as requirements specification, software design, implementation, testing, and so on.



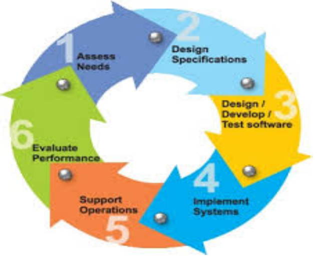
Waterfall Model (Cont..)

- **The sequential phases in Waterfall model are:**
  - ***Requirement Gathering and analysis:*** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
  - ***System Design:*** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
  - ***Implementation:*** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
  - ***Integration and Testing:*** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
  - ***Deployment of system:*** Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.
  - ***Maintenance:*** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.



Waterfall Model (Cont..)

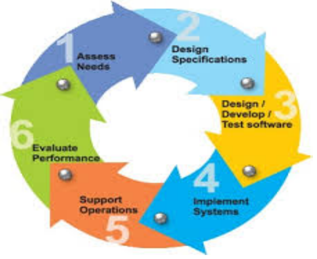
- **Problems:**
  - Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.
  - It is often difficult for the customer to state all requirements explicitly. The waterfall model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.
  - The customer must have patience. A working version of the program(s) will not be available until late in the project time span. A major blunder, if undetected until the working program is reviewed, can be disastrous.



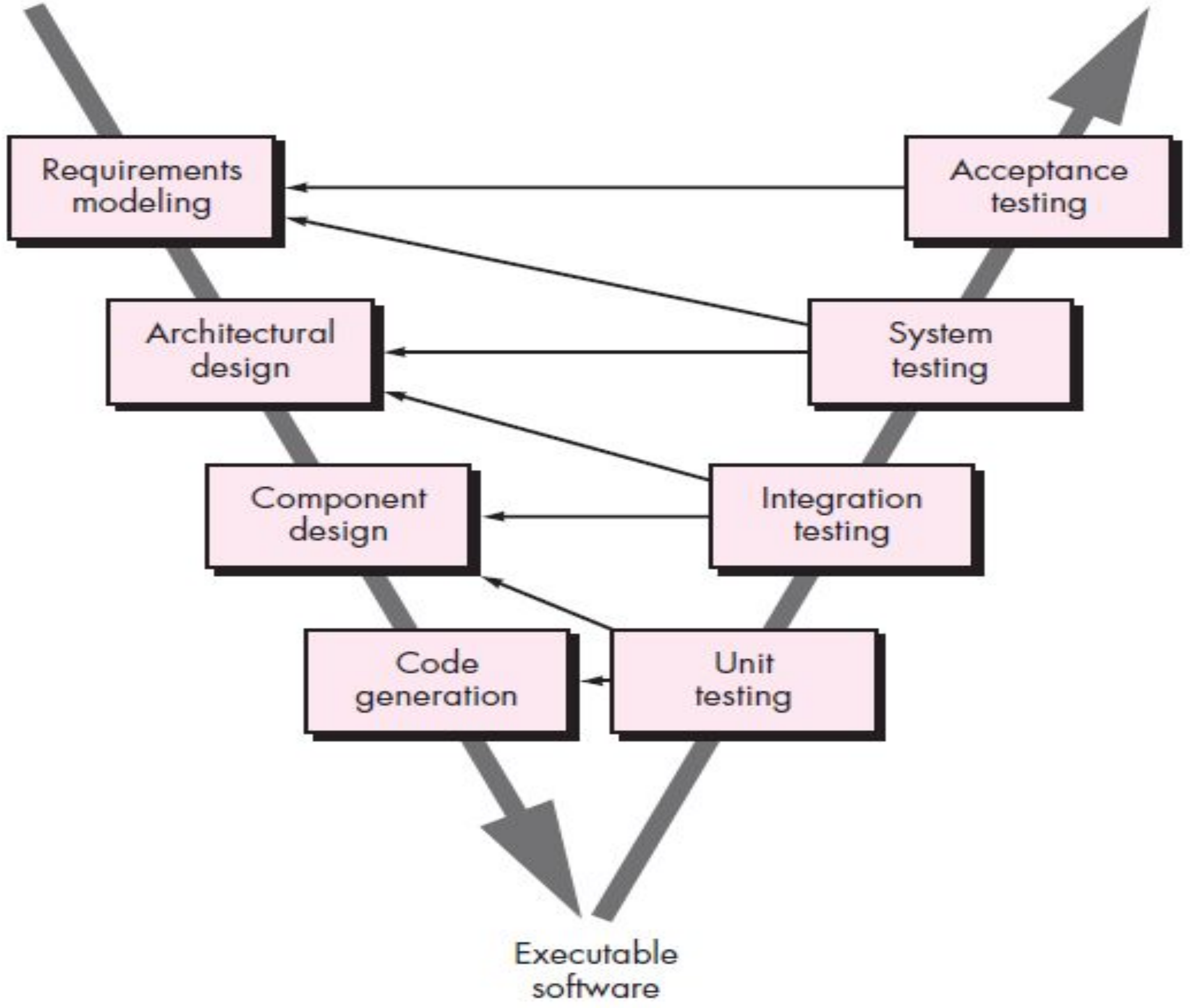
Waterfall Model (Cont..)

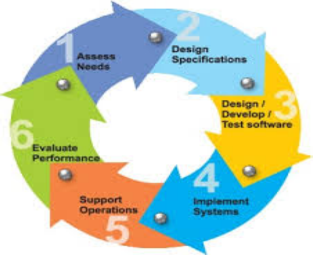
- **Problems:**
  - Rarely linear, iteration needed.
  - Hard to state all requirements explicitly. Blocking state.
  - Code will not be released until very late.

- **Waterfall Model Application**
  - Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:
  - Requirements are very well documented, clear and fixed.
  - Product definition is stable.
  - Technology is understood and is not dynamic.
  - There are no ambiguous requirements.
  - Ample resources with required expertise are available to support the product.
  - The project is short.


1 Assess Needs
2 Design Specifications
3 Design / Develop / Test software
4 Implement Systems
5 Support Operations
6 Evaluate Performance

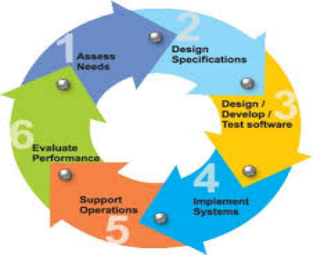
The V-Model


Requirements modeling
Architectural design
Component design
Code generation
Acceptance testing
System testing
Integration testing
Unit testing
Executable software



The V-Model(Cont..)

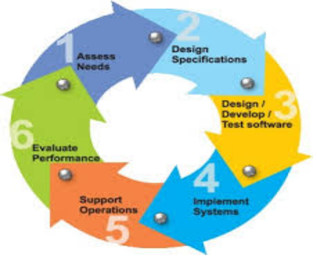
- The V - model is SDLC model where execution of processes happens in a sequential manner in V-shape.
- It is also known as ***Verification and Validation model***.
- V - Model is an extension of the waterfall model and is based on association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle there is a directly associated testing phase.
- This is a highly disciplined model and next phase starts only after completion of the previous phase.



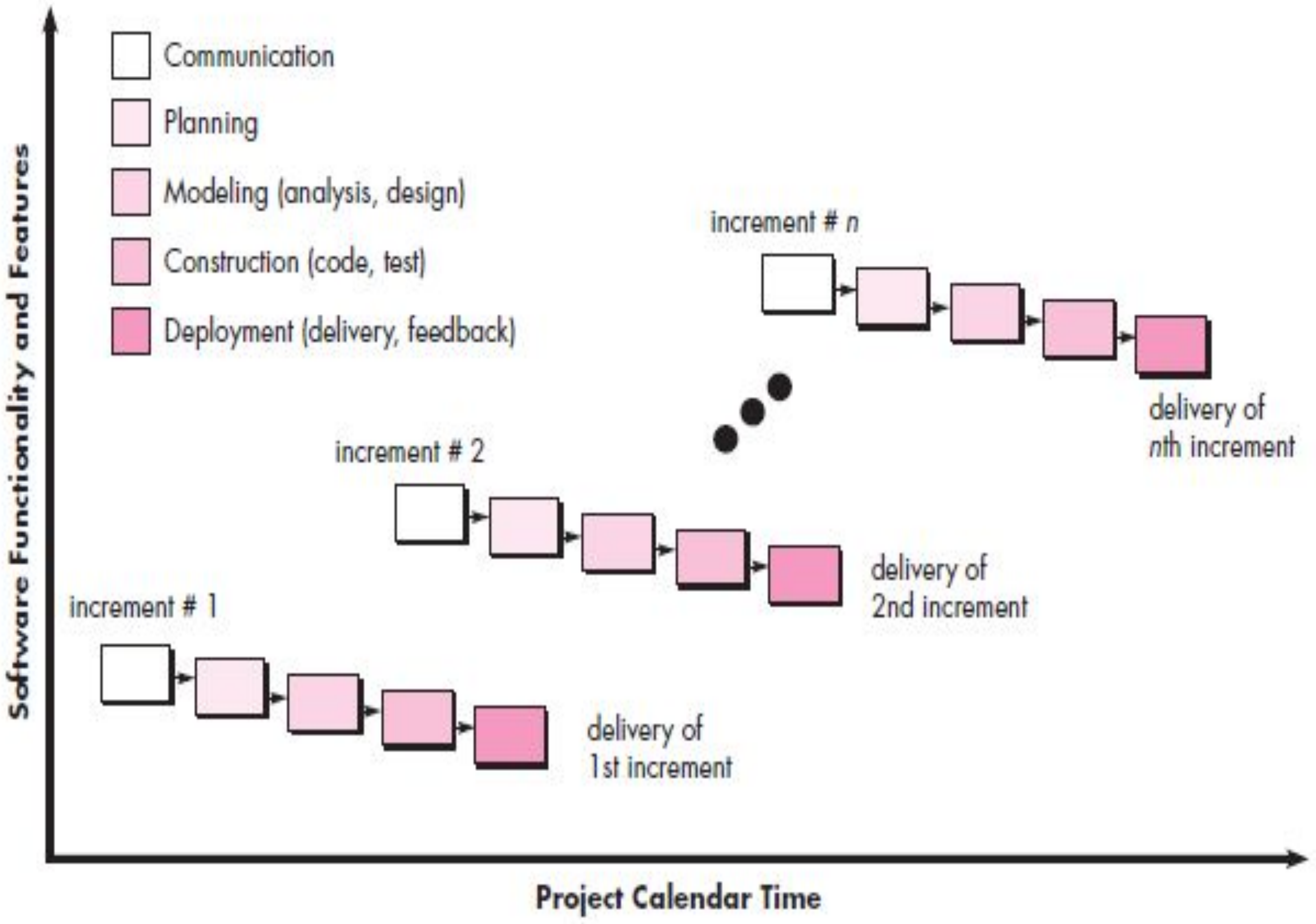
The V-Model(Cont..)

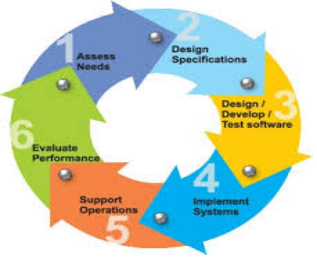
- A variation of waterfall model depicts the relationship of quality assurance actions to the actions associated with communication, modeling and early code construction activates.

- Team first moves down the left side of the V to refine the problem requirements. Once code is generated, the team moves up the right side of the V, performing a series of tests that validate each of the models created as the team moved down the left side.



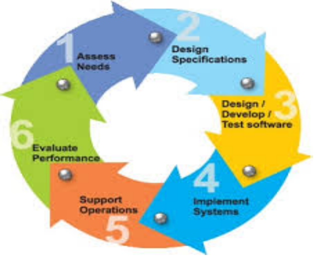
The Incremental Model


Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
increment # 1
delivery of 1st increment
increment # 2
delivery of 2nd increment
increment # n
delivery of nth increment
Software Functionality and Features
Project Calendar Time

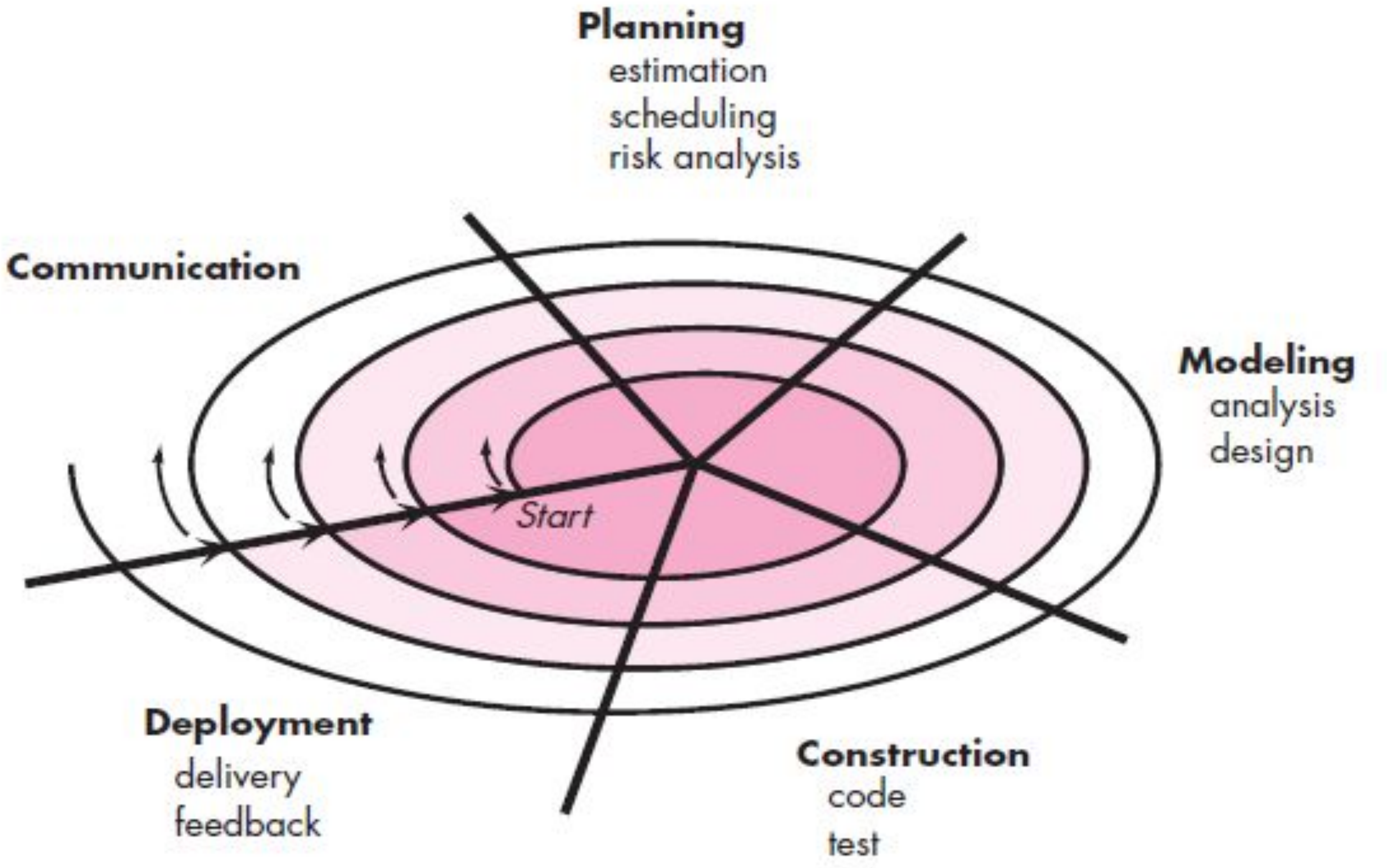


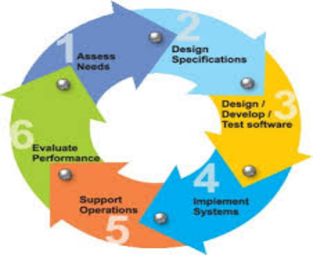
The Incremental Model(Cont..)

- The incremental model combines element of waterfall model applied in an iterative fashion.
- The incremental model applies linear sequence like as calendar time progresses. Each linear sequence.
- **For Example:-**
  - Word processing software development using the incremental paradigm.
  - Basic file management, editing, document production function in the first increment.
  - More sophisticated editing, and document production capabilities in the second increment.
  - Spelling and grammar checking in the third increment.
  - Advance page layout capability in the fourth.



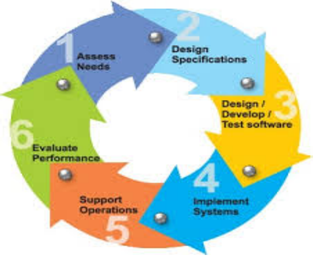
The Spiral Model





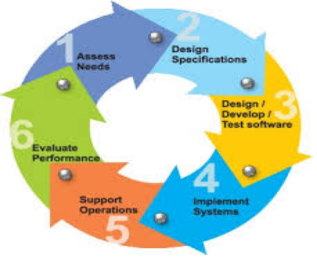
The Spiral Model(Cont..)

- The spiral model is a evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspect of the waterfall model.
- It provides the potential for rapid development of increasingly more complete version of the software.
- A spiral model is divided into a set of framework activities defined by software engineering team. Each framework activities represent one segment of the spiral path.
- Activities are implied by a circuit around the spiral clockwise direction, beginning at the center.

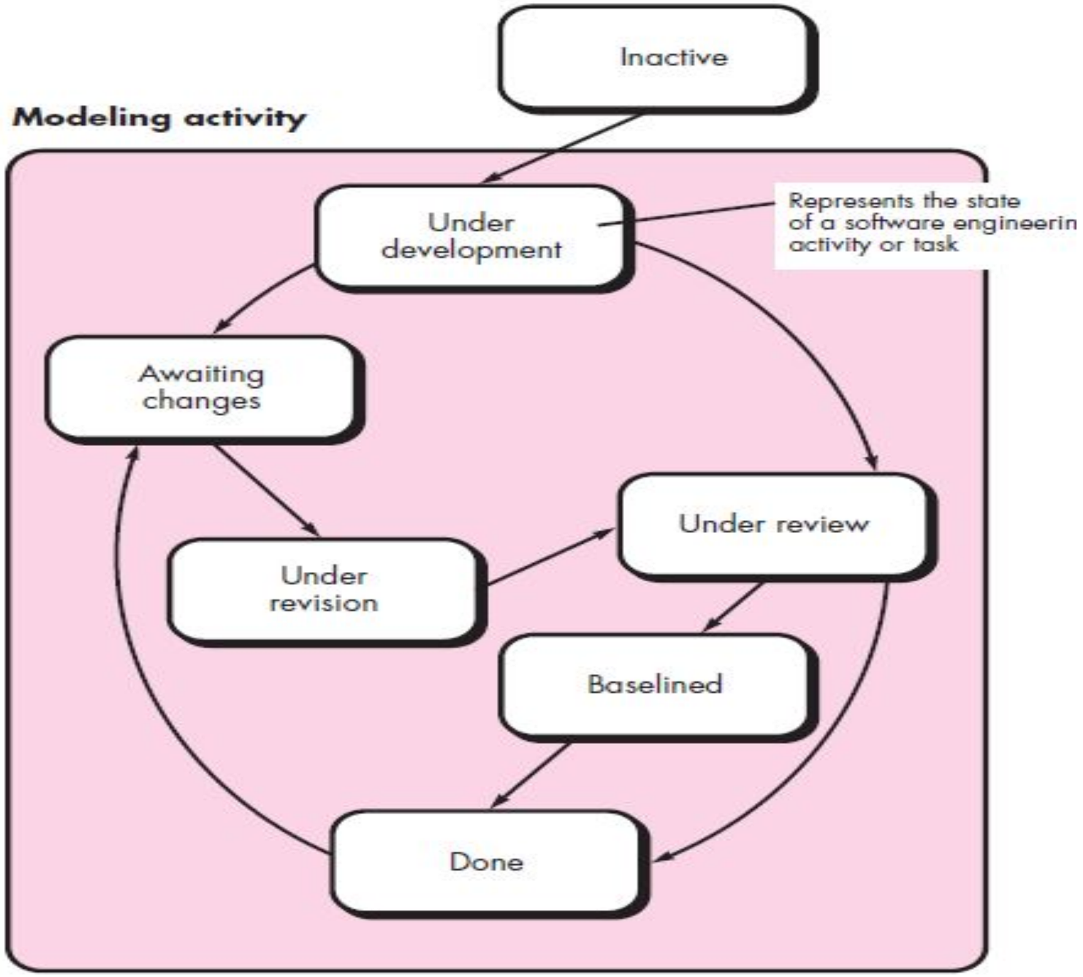


The Spiral Model(Cont..)

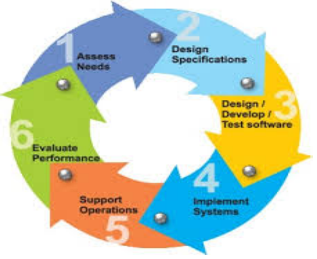
- Concept development project
  - Which starts at the core of the spiral and continues for multiple iteration.
- New product development
  - The new product will develop through a number of iteration around the spiral.
- The spiral model is a realistic approach to development of large scale system and software. Because software development the process progresses.



Concurrent Models

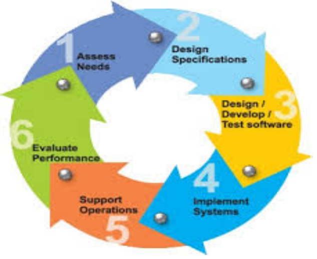


- **All software engineering activities exist concurrently but reside in different states.**
- For example, early in a project the communication activity (not shown in the figure) has completed its first iteration and exists in the awaiting changes state.



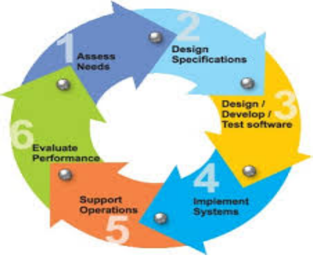
Concurrent Model(Cont..)

- **The concurrent development model -** called concurrent engineering
  - It provides an accurate state of the current state of a project.
  - Focus on concurrent engineering activities in a software engineering process such as prototyping, analysis modeling, requirements specification and design.
  - Represented schematically as a series of major technical activities, tasks and their associated states.
  - Defined as a series of events that trigger transitions from state to state for each of the software engineering activities.



Concurrent Model(Cont..)

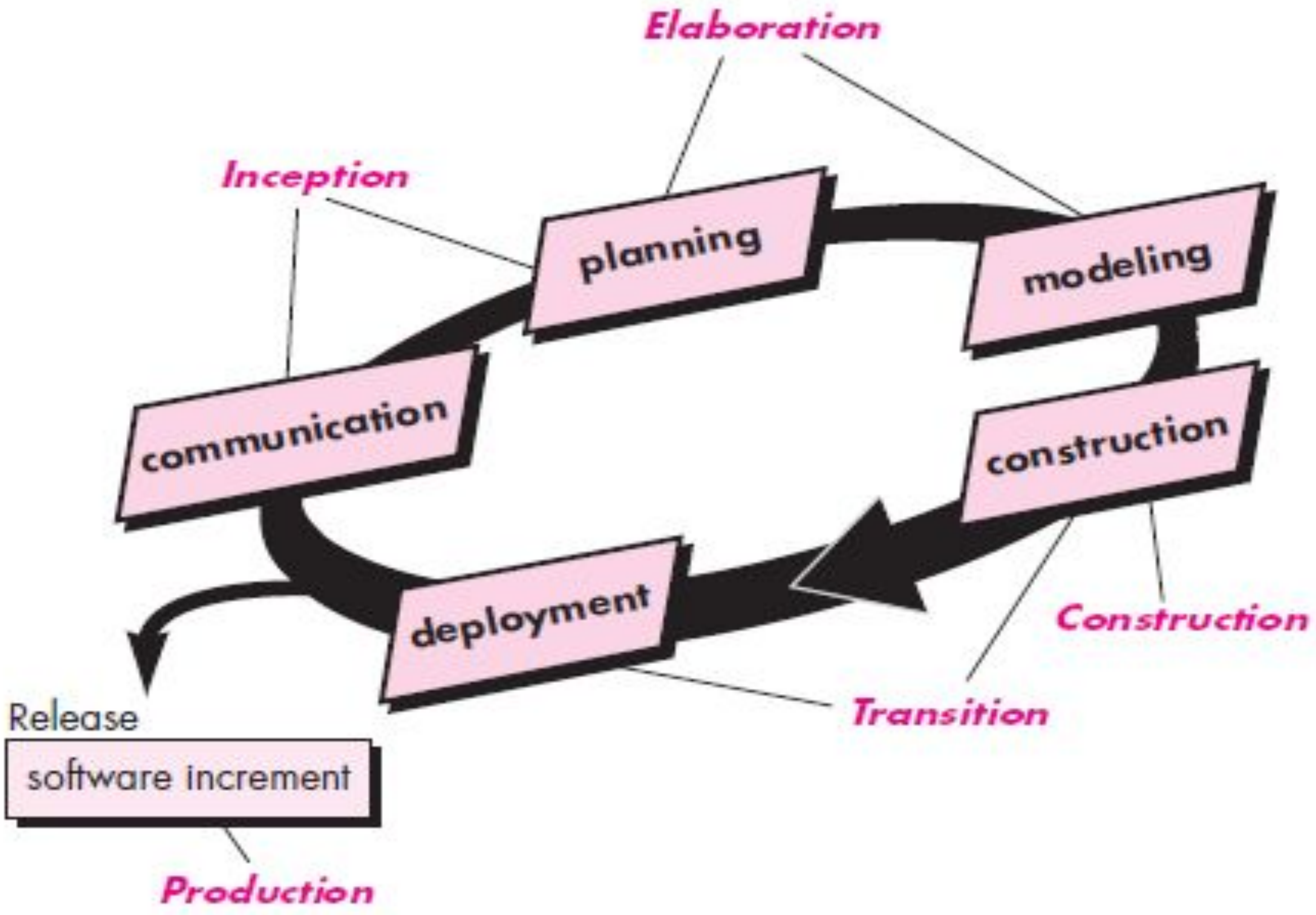
- Often used as the paradigm for the development of client/server applications. A client/server system is composed of a set of functional components.
- When applied to client/server, the concurrent process model defines activities in two dimensions :
  - (i) System dimension -System level issues are addressed using three activities: design, assembly, and use.
  - (ii) The component dimension is addressed with two activities: design and realization.

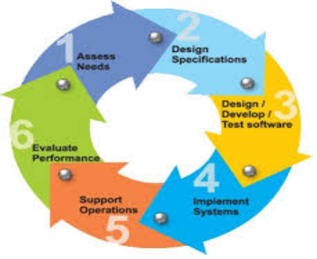


Unified Process Model

- Unified process (UP) is an **architecture-centric**, **use-case driven**, **iterative and incremental** development process that leverages unified modeling language and is compliant with the system process engineering meta-model.
- Unified process can be applied to different software systems with different levels of technical and managerial complexity across various domains and organizational cultures.
- UP is also referred to as the unified software development process.

The Unified Process (UP)





The Unified Process (UP)

Inception phase

Vision document
Initial use-case model
Initial project glossary
Initial business case
Initial risk assessment.
Project plan,
  phases and iterations.
Business model,
  if necessary.
One or more prototypes

Elaboration phase

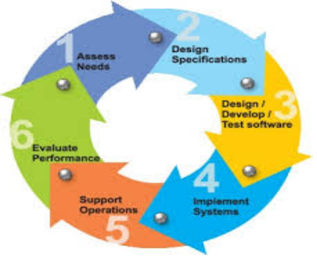
Use-case model
Supplementary requirements
  including non-functional
Analysis model
Software architecture
  Description.
Executable architectural
  prototype.
Preliminary design model
Revised risk list
Project plan including
  iteration plan
  adapted workflows
  milestones
  technical work products
Preliminary user manual

Construction phase

Design model
Software components
Integrated software
  increment
Test plan and procedure
Test cases
Support documentation
  user manuals
  installation manuals
  description of current
    increment

Transition phase

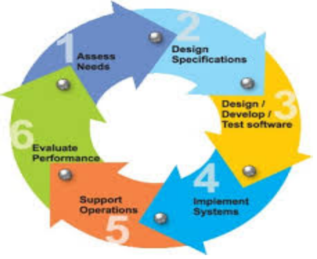
Delivered software increment
Beta test reports
General user feedback



Selection of a Life Cycle Model

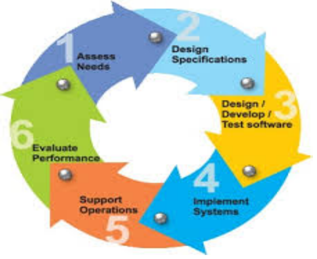
❑ Selection of a model is based on:

- Requirements
- Development team
- Users
- Project type and associated risk

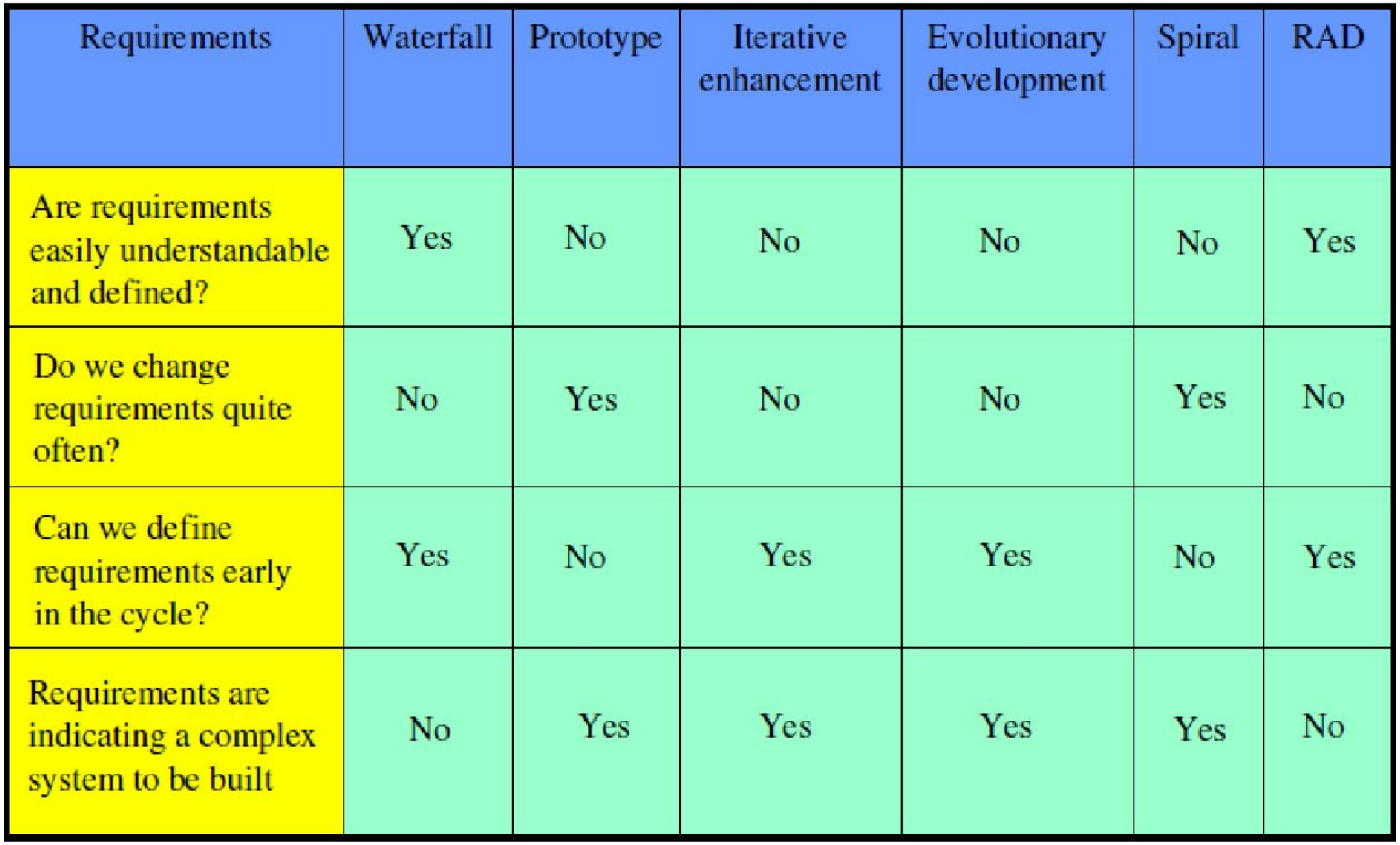


Selection of a Life Cycle Model

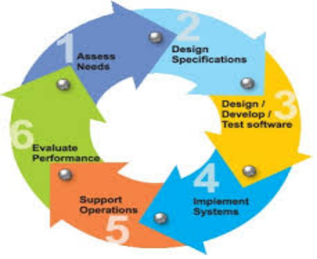
- Selection of a model is based on:
  - Requirements
  - Development team
  - Users
  - Project type and associated risk



Based On Characteristics Of Requirements

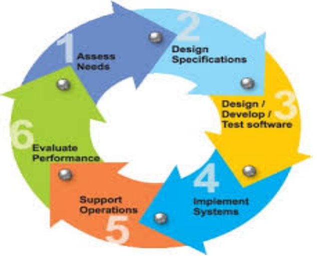


Requirements	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Are requirements easily understandable and defined?	Yes	No	No	No	No	Yes
Do we change requirements quite often?	No	Yes	No	No	Yes	No
Can we define requirements early in the cycle?	Yes	No	Yes	Yes	No	Yes
Requirements are indicating a complex system to be built	No	Yes	Yes	Yes	Yes	No



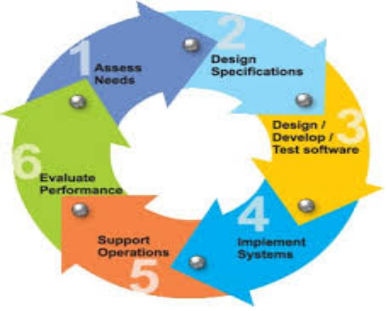
Based On Status Of Development Team

Development team	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Less experience on similar projects?	No	Yes	No	No	Yes	No
Less domain knowledge (new to the technology)	Yes	No	Yes	Yes	Yes	No
Less experience on tools to be used	Yes	No	No	No	Yes	No
Availability of training if required	No	No	Yes	Yes	No	Yes



Based On User's Participation

Involvement of Users	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
User involvement in all phases	No	Yes	No	No	No	Yes
Limited user participation	Yes	No	Yes	Yes	Yes	No
User have no previous experience of participation in similar projects	No	Yes	Yes	Yes	Yes	No
Users are experts of problem domain	No	Yes	Yes	Yes	No	Yes



❑ References:

1. **Software Engineering A practitioner's Approach**

   by Roger S. Pressman, 7th edition, McGraw Hill, 2010.

2. **Software Engineering by Ian Sommerville,**

   9th edition, Addison-Wesley, 2011

3. **Software Engineering Practices**

   Mohamed Sami, https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/