SOFTWARE ENGINEERING

UNIT – 1

By Tulasi Vemu.

INTRODUCTION TO SOFTWARE ENGINEERING

- Objective Of the Course SE
- What Is Software
- Introduction about Software

SOFTWARE

Software is

1. Instructions of a computer system, that when executed provides desired features/functions and performance.

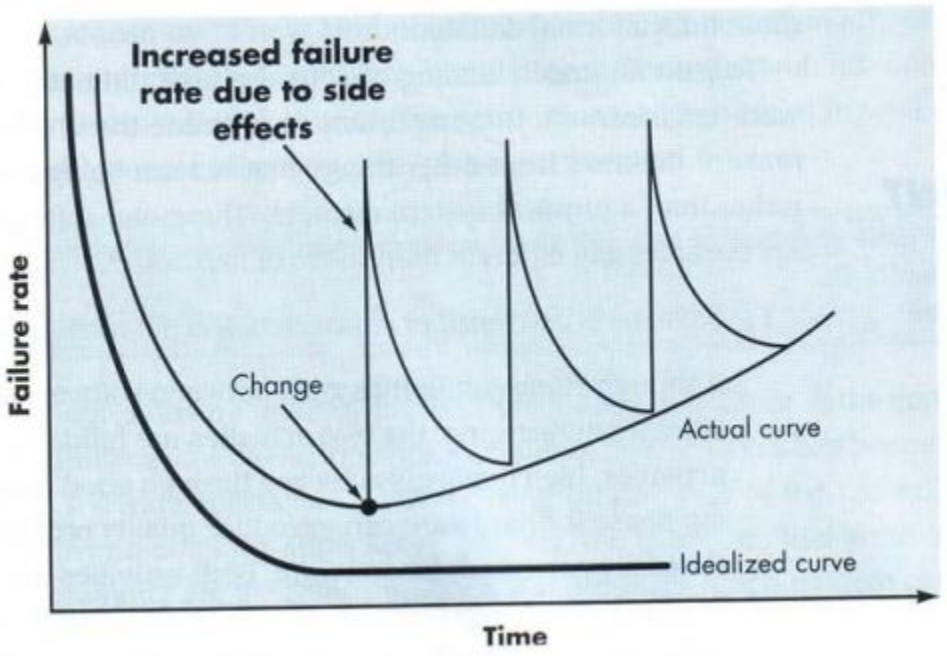
2. data structures that enables the program to adequately manipulate the information

3. Document that describes the operation and use of programs.

CHARACTERISTICS OF THE SOFTWARE

- Software is developed/ engineered but it is not manufactured in classical sence.
- Software doesn’t “wear out”.
- Although the industry is moving towards component based construction, most software continues to be custom-built.

FAILURE CURVES FOR SOFTWARE



NATURE OF THE SOFTWARE

- System Software
- Application Software
- Engineering /Scientific Software
- Embedded Software
- Product-line software
- Web Applications
- Artificial Intelligence Software
- Ubiquitous computing

THE SOFTWARE PROCESS

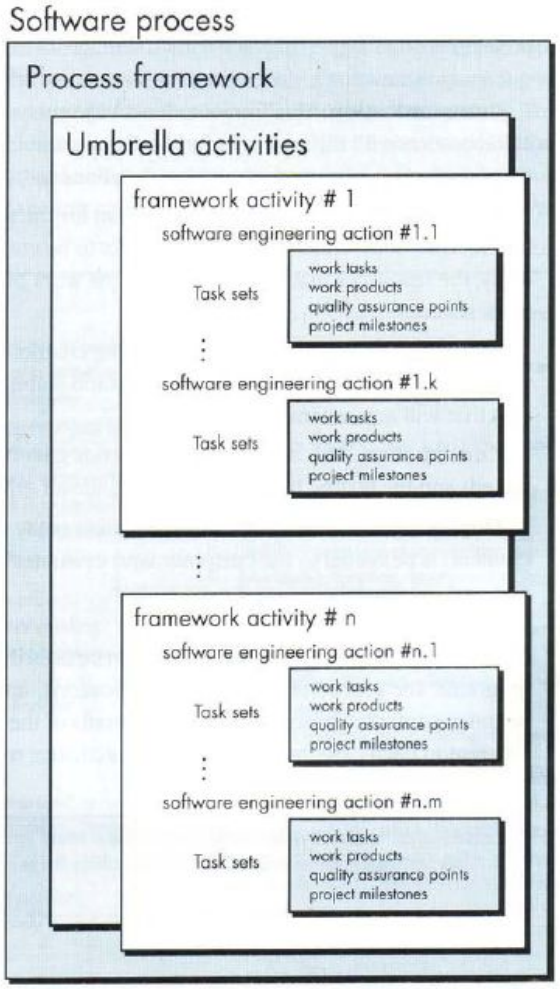
- What is it?
- What does it do?
- Why is it important?
- What are the steps?
- What is the work product?
- How do I ensure that I have done it right?

SOFTWARE ENGINEERING - LAYERED TECHNOLOGY

A GENERIC PROCESS MODEL

- Communication
- Planning
- Modeling
- Construction
- Deployment

SOFTWARE PROCESS FRAMEWORK



PROCESS FRAMEWORK…

- The Process Framework establishes the foundation for a complete software process by identifying small number of framework activities that are applicable to all software projects irrespective of size and complexity.

- This consists of set of umbrella activities that are applicable throughout entire process.

FRAMEWORK ACTIVITIES

- Communication
- Planning
- Modeling
- Construction
- Deployment

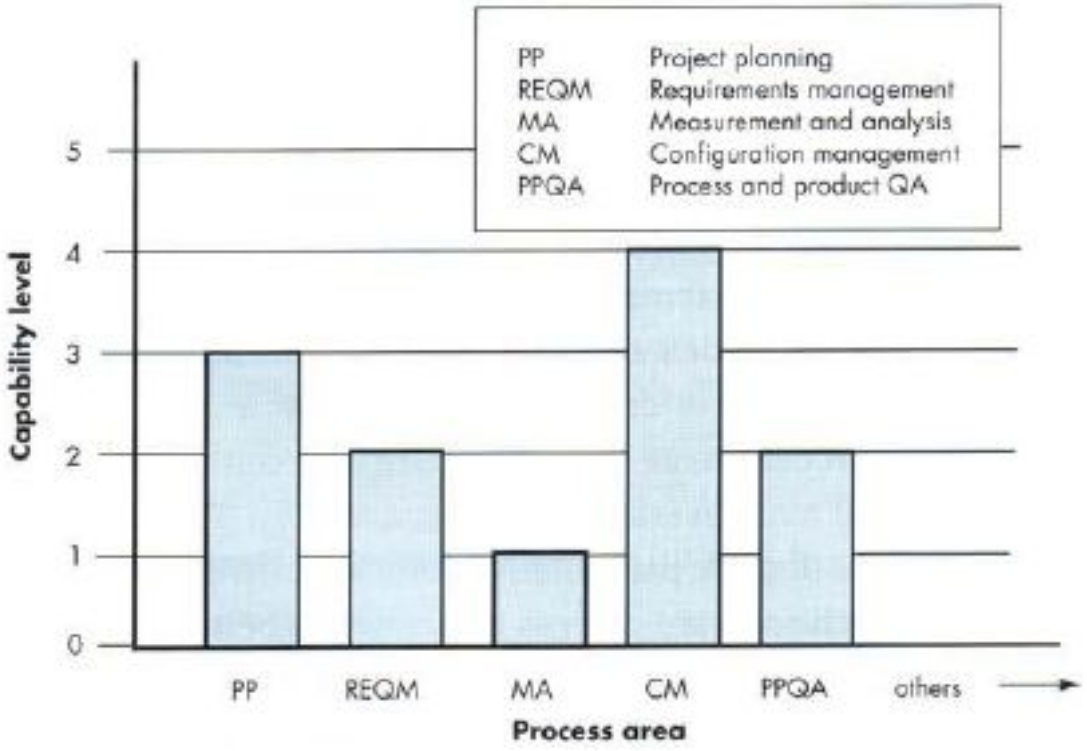
SOFTWARE ENGINEERING ACTIVITIES

- Modeling
  - Analysis
    - Requirements Gathering
    - Elaboration
    - Negotiation
    - Specification
    - validation
  - Design
    - Data Design
    - Architectural design
    - Interface Design
    - Component Level Design

UMBRELLA ACTIVITIES

- Software Project Tracking and control
- Risk Management
- Software Quality assurance
- Formal Technical Reviews
- Measurement
- Software Configuration Management
- Reusability management
- Work product preparation and production

THE CAPABILITY MATURITY MODEL INTEGRATION (CMMI)



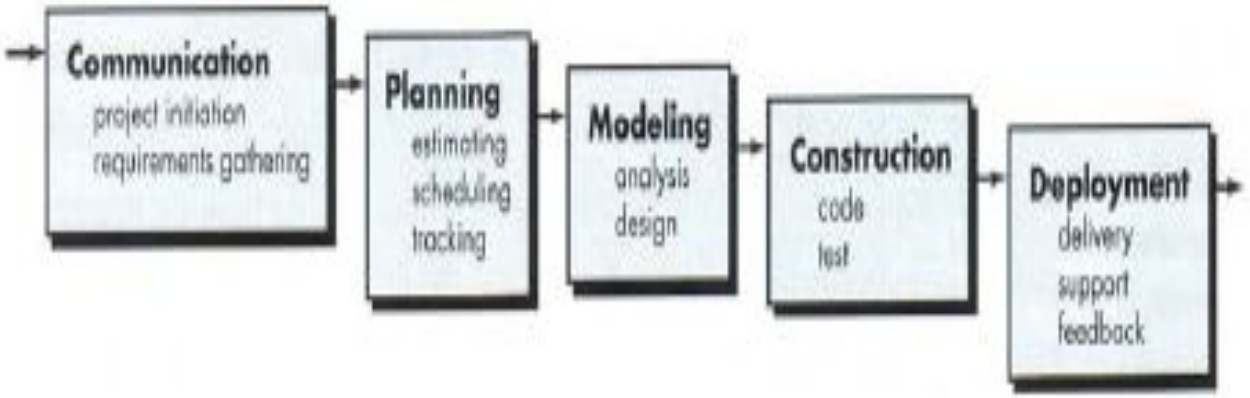
CMMI LEVELS

- Level 0: Incomplete
- Level 1: Performed
- Level 2: Managed
- Level 3: Defined
- Level 4: Quantitatively managed
- Level 5: Optimized

THE SOFTWARE PROCESS MODEL

- The Waterfall Model
- The Incremental Process Models
  - The incremental Model
  - The RAD (Rapid Application Development) Model
- The Evolutionary Process Models
  - The Spiral Model
  - The Prototyping Model
  - The Concurrent Development Model
- The specialized Process Models
  - The Component Based Model
  - The Formal Methods
  - The Aspect-Oriented Software
- The Unified Process Model

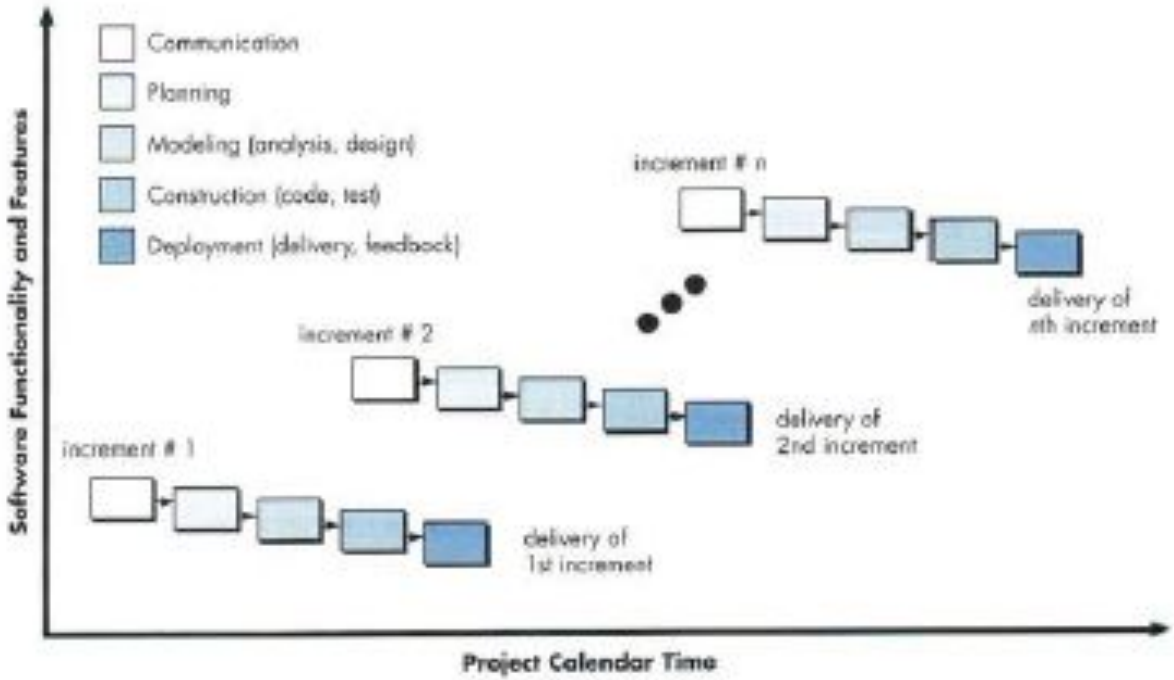
THE WATERFALL MODEL



ADVANTAGES & DISADVANTAGES

- Advantages
  - This is useful when well-defined adoptions or enhancements to be made on an existing system.
  - It Follows a systematic, sequential Approach
- Disadvantages
  - Real Projects rarely follow the sequential flow that the model possess.
  - It is often difficult for the customers to state all requirements Explicitly.
  - Customers should wait with patience until the end the project as it takes longer time than expected.

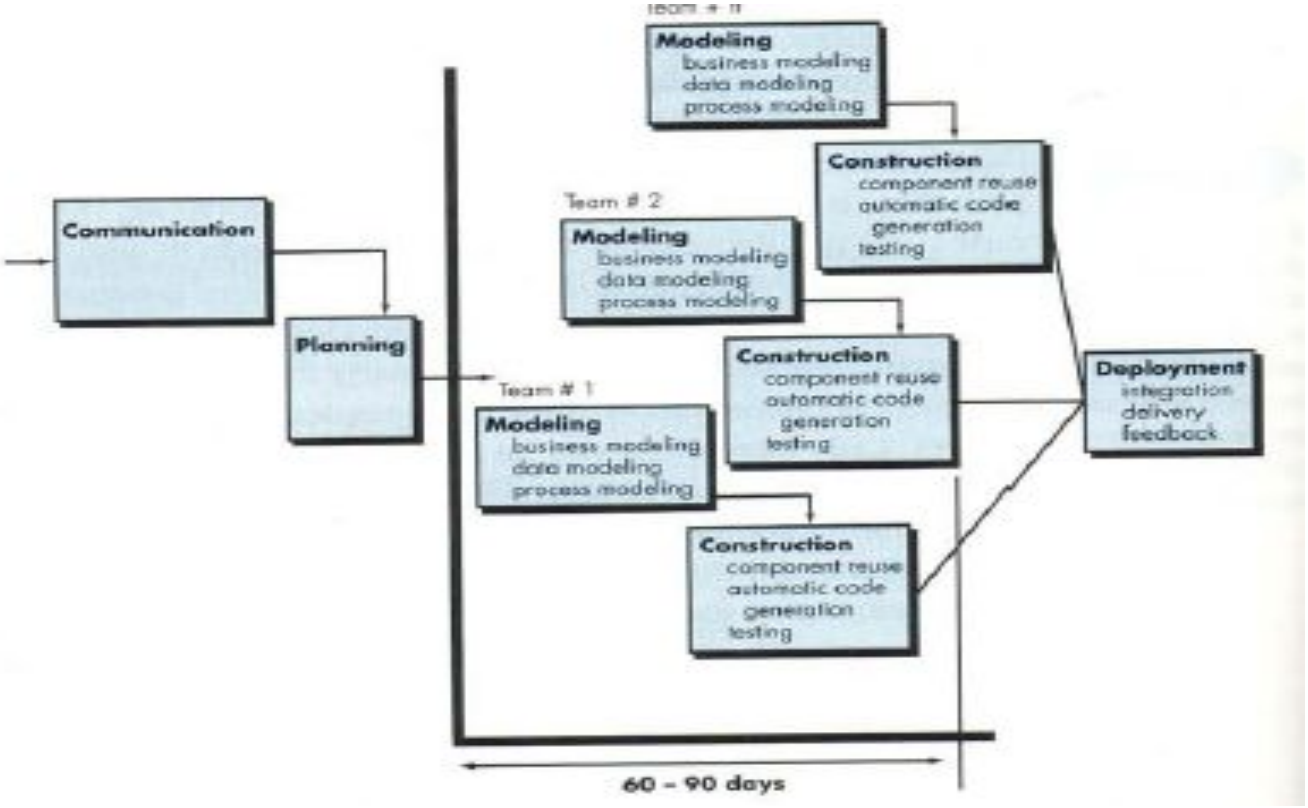
THE INCREMENTAL MODEL



ADVANTAGES & DISADVANTAGES

- Advantages
  - Focuses on delivery of an operational Model at each increment.
  - Particularly useful when staffing is unavailable.
  - Increments can be planned to manage technical risks.
- Disadvantages
  - Needs good planning and design.
  - Needs a clear and complete definition of the whole system before it can be broken down and built **incrementally**.
  - Total cost is higher than waterfall.

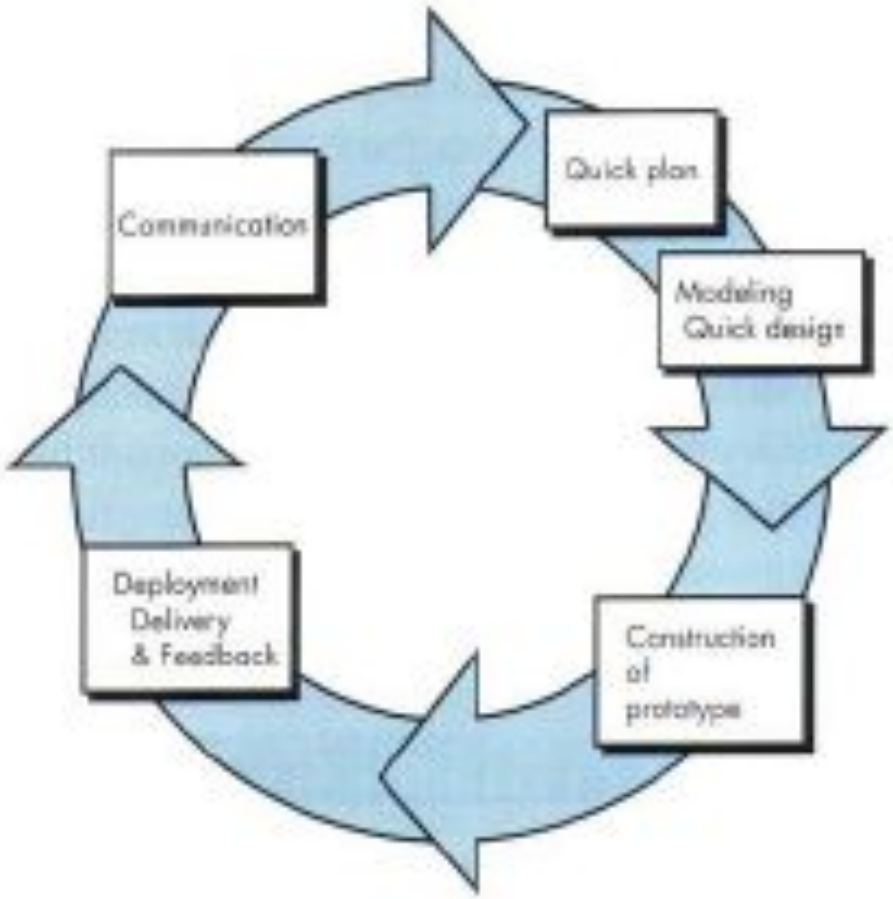
THE RAD MODEL



ADVANTAGES & DISADVANTAGES

- Advantages
  - Emphasizes on short development life cycle.(Less time required to finish the project).
  - Fully Functional system will be developed in short period of time.
- Disadvantages
  - More people required to form sufficient RAD teams.
  - If developers or customers doesn't perform Rapid Fire actions, then system will fail.
  - If the system was not properly modularized, then building the components will be problematic.
  - With RAD High performance may not be achieved.
  - RAD may not be appropriate with the projects having high technical risks.

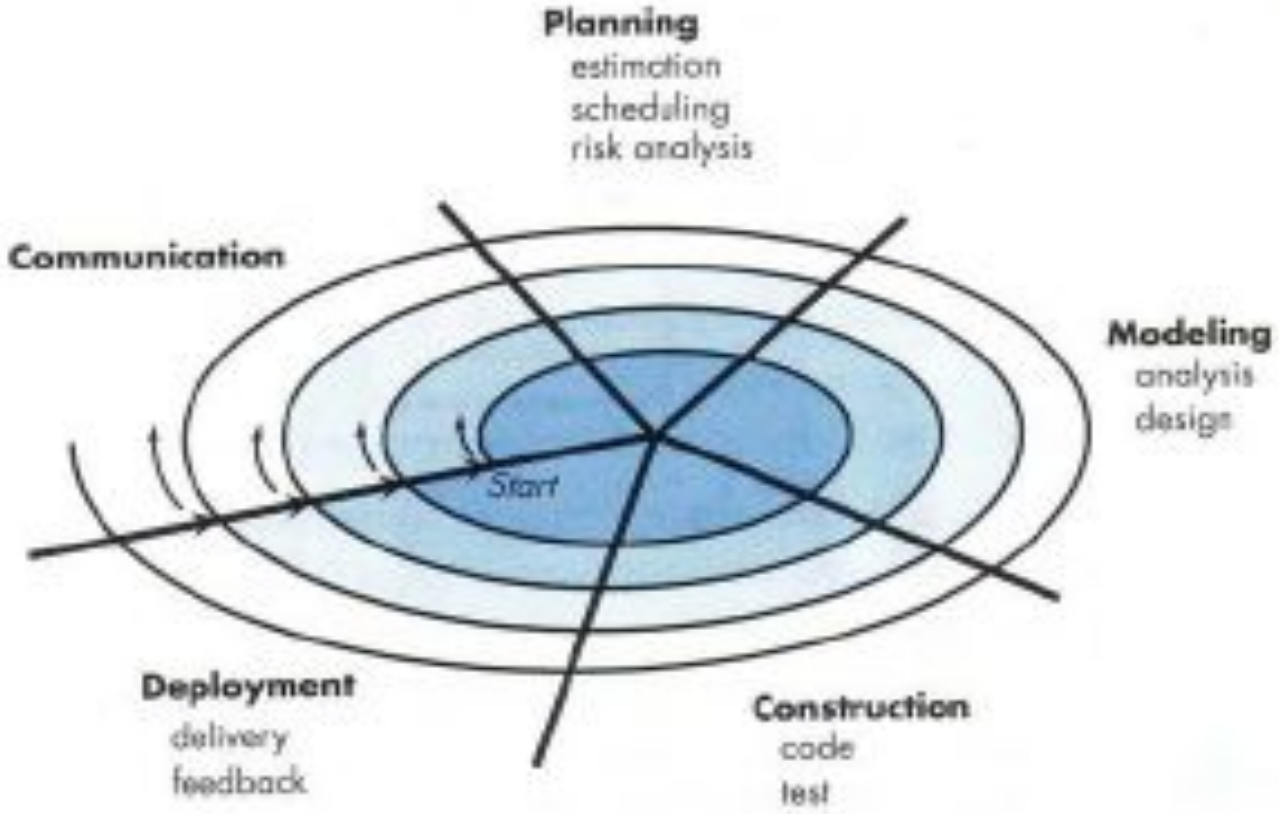
THE PROTOTYPING



ADVANTAGES & DISADVANTAGES

- Advantages
  - This is Iterative in natural.
  - Best suitable for the projects which are defined with general objectives and doesn’t identify the detailed input/processing/output requirements.
- Disadvantages
  - May not achieve overall software quality & Long term maintainability.
  - Developer may make implementation compromises in order to work prototype quickly.

THE SPIRAL MODEL



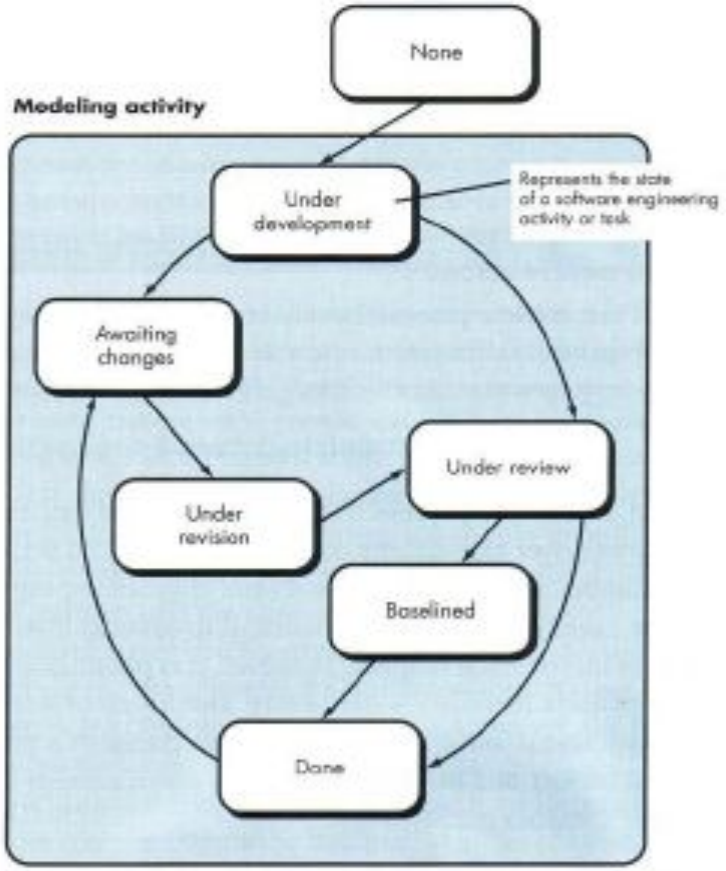
ADVANTAGES & DISADVANTAGES

- Advantages
  - Risk handling is one of important advantages of the Spiral model, it is best development model to follow due to the risk analysis and risk handling at every phase.
  - Flexibility in requirements. In this model, we can easily change requirements at later phases and can be incorporated accurately. Also, additional Functionality can be added at a later date.
  - It is good for large and complex projects.
  - It is good for customer satisfaction. We can involve customers in the development of products at early phase of the software development. Also, software is produced early in the software life cycle.

ADVANTAGES & DISADVANTAGES

- Disadvantages
  - It is not suitable for small projects as it is expensive.
  - It is much more complex than other SDLC models. Process is complex.
  - Too much dependable on Risk Analysis and requires highly specific expertise.
  - Difficulty in time management. As the number of phases is unknown at the start of the project, so time estimation is very difficult.
  - Spiral may go on indefinitely.
  - End of the project may not be known early.

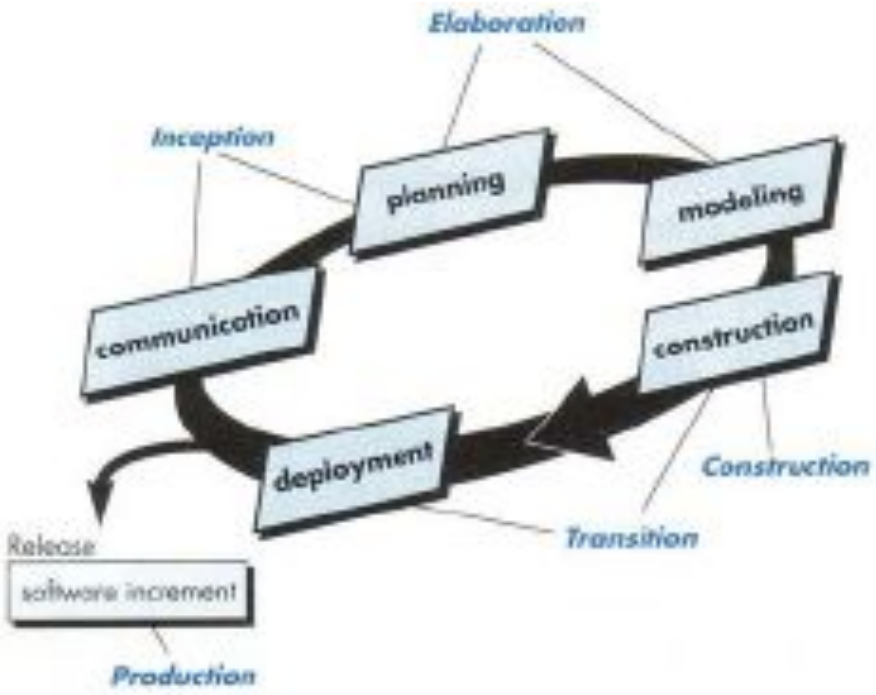
THE CONCURRENT DEVELOPMENT MODEL



ADVANTAGES & DISADVANTAGES

- Advantages
  - The Process model is applicable to all types of projects and provides an accurate picture of the current state of the project
- Disadvantages
  - Creates problem to project planning because of the uncertainty in the no. Of cycles required to complete the development phase.

THE UNIFIED PROCESS MODEL



PHASES OF UNIFIED PROCESS MODEL

- Inception
- Elaboration
- Construction
- Transition
- Production

INCEPTION

- This phase Consists of Customer communication and planning activities
- Fundamental business requirements are described through a set of preliminary use cases which describes the features and functionalities.
- Architecture at this point is a tentative outline of major subsystems and functions/ features that populate them.

ELABORATION

- This phases encompasses of planning and modeling activities.
- Elaboration refines and expands preliminary use cases and also expands the architectural representation to include five different views of the software – the use-case model, the analysis model, the design model , the implementation model and deployment model.
- The plan is carefully reviewed to ensure the scope, risk and delivery dates to be reasonable.

CONSTRUCTION

- The construction phase is similar to the construction activity defined for the generic process model.
- All required features, functions of that incremented are implemented as source code.
- Unit test are been executed on each component.
- Integration activities and integration testing is carried Out.

TRANSITION & PRODUCTION

- Latter activities of construction phase, Early stages of deployment phase of generic process model.
- Software is given to the end users for beta testing.

- Giving Final Project to the customers.

THE SPECIALIZED PROCESS MODELS

- The Component Based Model
- The Formal Methods
- The Aspect-Oriented Software

THE COMPONENT BASED MODEL

- Commercial – Off – The _Shelf (COTS) components are developed by the vendors, who offer them as products are used by the software developers to build their software.

- These Components provide targeted Functionality with well defined interfaces that enables the component to be integrated into the software.

THE COMPONENT BASED MODEL

- Available Component based products are researched and evaluated for the application domain in the question.
- Component integration issues are considered.
- A software architecture is designed to accommodate the components.
- Components are integrated to the architecture.
- Comprehensive testing is conducted to ensure the functionality.

THE FORMAL METHODS

- This method consists of set of activities that leads to the formal mathematical specification of the computer software.
- It enables the software engineer to specify, develop and verify a computer based system by applying a rigorous mathematical notation.
- This method provides a mechanism to eliminate many problems that are difficult to overcome them using other software development models.

THE FORMAL METHODS

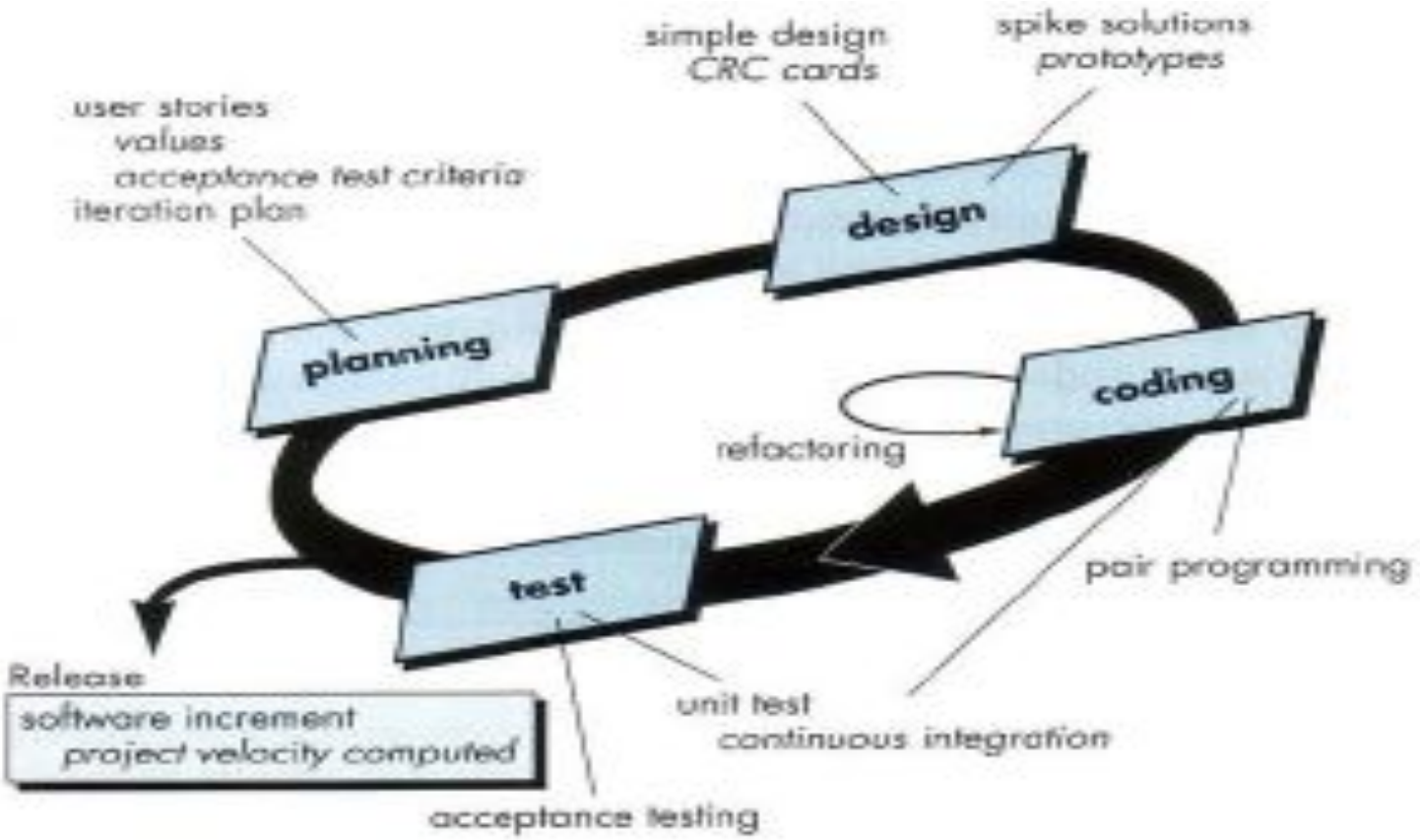
- Concerns
  - The development of formal methods are quite time consuming and expensive.
  - Only few software engineers are have knowledge of formal methods, it is required to train the people when you choose this method.
  - In this method communication mechanism is unsophisticated to the customers.

THE ASPECT ORIENTED SOFTWARE

Aspectual Requirements are verified across the Life Cycle Phases.

Practical Limitations Due to Technical Aspects are considered as Cross cut Requirements or Aspectual Requirements.

AGILE METHODOLOGY



PROCESS & PRODUCT

QUICK LOOK

What is it? When you work to build a product or system, it's important to go through a series of predictable steps—a road map that helps you create a timely, high-quality result. The road map that you follow is called a *software process*.

Who does it? Software engineers and their managers adapt the process to their needs and then follow it. In addition, the people who have requested the software have a role to play in the process of defining, building, and testing it.

Why is it important? Because it provides stability, control, and organization to an activity that can, if left uncontrolled, become quite chaotic. However, a modern software engineering approach must be "agile." It must demand only those activities, controls, and documentation that are appropriate for the project team and the product that is to be produced.

What are the steps? At a detailed level, the process that you adopt depends on the software that you're building. One process might be appropriate for creating software for an aircraft avionics system, while an entirely different process would be indicated for the creation of a Web site.

What is the work product? From the point of view of a software engineer, the work products are the programs, documents, and data that are produced as a consequence of the activities and tasks defined by the process.

How do I ensure that I've done it right? There are a number of software process assessment mechanisms that enable organizations to determine the "maturity" of their software process. However, the quality, timeliness, and long-term viability of the product you build are the best indicators of the efficacy of the process that you use.

PERSONAL SOFTWARE PROCESS

- Planning
- High Level design
- High Level design Review
- Development
- Postmortem

TEAM SOFTWARE PROCESS

- Build self-directed teams that plan and track their work, establish goals, and own their processes and plans. These can be pure software teams or integrated product teams (IPT) of 3 to about 20 engineers.
- Show managers how to coach and motivate their teams and how to help them sustain peak performance.
- Accelerate software process improvement by making CMM level 5 behavior normal and expected.
- Provide improvement guidance to high-maturity organizations.
- Facilitate university teaching of industrial-grade team skills.