**Chapter 2 Notes**

**Chapter 2**

**SW Process**: Set of related activities that leads to the production of a software product.

1. Specification
2. Development
3. Validation
4. Evolution

**2 Different Categories for SW Processes**:

1. Plan-driven
   1. All of the process activities are planned in advance and progress is measured against this plan
2. Agile Processes
   1. Planning is incremental and it is easier to change the process to reflect changing customer requirements.

**Process Models**

1. Waterfall Model
   1. Takes the fundamental process activites of specification, development, validation, and evolution and represents them as process pahses such as:
      1. Requirements Definition
      2. System and Software Design (design the entire architecture)
      3. Implementation and Unit Testing
      4. Integration and System Testing (all the programs are integrated and tested)
      5. Operation and Maintainance (put into use in market and any bugs are corrected)
   2. It is Plan-driven
   3. Each phase depends on the previous phase
   4. Weakness: its inflexible to partition the project into distinct phases, commitments must be made at an early stage in the process.
   5. This should only be used when requirements are well understood and unlikely to change radically during development.
2. Incremental Developments
   1. Interleaves the activities of specification, development, and validation. The systems are developed as a series of version (increments), with each version adding functionality to the previous version
   2. Is Agile-Driven
   3. Developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate system has been develop.
   4. Specification, Development, and Validation activities are interleaved rather than separate.
   5. Advantages: Cheaper and easier to make changes, don't have to define all requirements in the beginning, rapid delivery, and easier to get customer feedback
   6. Disadvantages: It is not feasible to create documents for each release and System structure tends to degrade as new incremented are added
3. Reuse-oriented Software Engineering
   1. Based on the existence of a significant number of reusable components. The SW development process focuses on integrating these components into a system rather than developing them from scratch.
   2. Phases
      1. Requirements Specification
      2. Component Analysis
         1. Search for component to implement the specification
      3. Requirements Modification
         1. Requirements modified to suit the component analyzed
      4. System Design with Reuse
         1. System is designed or an existing framework is reused
      5. Development and Integration
      6. System Validation
4. Examples of Reusable Components
   1. Web services
   2. Packages in .NET or j2EE
   3. Stand-alone systems
5. Advantages
   1. Reduces the amount of software to be developed and reducing costs and risks
6. Disadvantages
   1. Control is lost over components that are reused

**Process Activities**

1. SW Specification
   1. The process of understanding and defining what services are required from the system and identifying the constraint on the systems operation and development
   2. Presented at 2 levels
      1. End-users and Customer
      2. Developers
   3. 4 Main Activities
      1. Feasibility study
         1. Decides whether it is feasible to go ahead with a more detailed analysis
      2. Requirements elicitation and analysis
         1. Other existing systems are inspected and discussions with potential users occur here
      3. Requirements specification
         1. Translating the information gathered in previous stage into a document (1 is User Requirements for the customer and 1 is System Requirements for the developers)
      4. Requirements Validation
         1. Requirements are checked for realism, consistency, and completeness.
2. Software Design and Implementation
   1. Design Inputs
      1. Platform Information
      2. Requirements Specification
      3. Data Description
   2. Design Activities
      1. Architectural Design
         1. Design the overall structure of the system and the components it will use and their relationships
      2. Interface Design
         1. Define the interface between components
      3. Component Design
         1. Design how each component will operate
      4. Database Design
         1. Design system data structures and how to represent these in a DB
   3. Design Outputs
      1. System Architecture
      2. Database Specification
      3. Interface Specification
      4. Component Specification
3. Software Validation
   1. Goal: To show that the system both conforms to specification and that it meets the customer's expectations
   2. Activities
      1. Component Testing
         1. Each component is tested independently without other system compontents by the people developing the system.
      2. System Testing
         1. Components are integrated and testing is done to find errors that result from interactions between components and component interface problems
      3. Acceptance Testing
         1. System is tested with data supplied by the system custome
4. Software Evolution
   1. Define System Requirements
   2. Assess Existing Systems
   3. Propose System Changes
   4. Modify Systems or Create New System

**2 Approaches to Reduce Rework Costs**

1. Change Avoidance (by using Prototypes)
2. Change Tolerance
3. 2 Ways of coping with change and changing system requirements
   1. System Prototyping -> a version or part of the system is developed quickly to check the customer's requirements and the feasibility of some design decisions
      1. This is Change Avoidance b/c it allows users to experiment with the system before delivery and therefore refine their requirements. (The number of requirement change proposals made after delivery is therefore likely to be reduced)
   2. Incremental Delivery ->System increments are given to customer for comment and experimentation
      1. This is Change Avoidance and Change Tolerance.
      2. Avoids premature commitment to requirements for the whole system and allows changes to be incorporated into later increments at low cost

**Prototyping Processing**

1. Establish Prototype Objectives (creates Protoyping Plan)
2. Define Protoype Functionality (creates Outline Definition)
3. Develop Prototype (creates Executable Protoype)
4. Evaluate Prototype (creates Evaluation Report)

**Prototyping**

1. Helps validate requirements
2. Helps explore SW solutions and GUI design

**Incremental Delivery Process**

1. Define Outline Requirements
2. Assigne Requiremtns to Increments
3. Design System Architexture
4. Develop System Incremental
5. Validate Incremental
6. Integrate Incremental
7. Validate Systematic
8. Deploy Increment and maybe begin developing next increment

**Incremental Delivery**

1. Advantages
2. Customer can gain experience with system in increments and learn about more requirements for later increments
3. Because high-priority services are delivered first, these will be tested heavily at first and also by each increment
4. Disadvantages
   1. Difficult to identify common facilities that are needed by all increments

**Spiral Model**

1. 4 stages
   1. Object Setting
      1. Determine Object, the potential alternatives, and the constraints on the system
   2. Risk Assessment and Reduction
      1. For each risk identified, do a detailed analysis.
      2. Evaluate alternatives and resolve risks and reduce them
   3. Development and Validation
      1. A development model is chosen (i.e. Waterfall, Increment, etc.)
      2. Develop and Verify product
   4. Planning
      1. Review project and decide whether to continue another iteration of the loop
      2. Plan next phase
2. THIS COMBINES CHANGE AVOIDANCE AND CHANGE TOLERANCE

**Rational Unified Process (RUP) (is a Process Model)**

1. 3 Perspectives
   1. Dynamic -> Shows the phases of the model over time
   2. Static -> shows the process activities that are enacted
   3. Practice -> Suggests good practices to be used during the process
2. Phases (Dynamic)
   1. Inception
      1. Establish a business case for the system
      2. Define who will interact with the system and how they will interact
      3. Evaluate the systems contribution to business
   2. Elaboration
      1. Develop and understand the problem domain
      2. Establish the architectural framework of the systematic
      3. Develop project plan and identify project risks
      4. Have requirements for the systematic
   3. Construction
      1. Design, programming, and testing
      2. Integrate systems
   4. Transition
      1. Move system into user community and make it work
3. Workflows (Static)
   1. These are not mapped to a single phase but can exist in whatever phase
   2. List
      1. Business Modeling
      2. Requirements
      3. Analysis and Design
         1. Design model is create and documented using Architectural, Component, Object, and Sequence Models
      4. Implementation
      5. testing
      6. Deployment
      7. Configuration and Change Management
         1. Manages changes to system
      8. Project Management
         1. Manage system development
      9. Environment
         1. Make appropriate SW tools available to the software team
4. Practice Perspective
   1. Develop SW Iteratively
      1. Plan increments and develop high-priority component first
   2. Manage requirements
   3. Use component-based architectures
   4. Visually model software
   5. Verify SW Quality
   6. Control Changes to SW