Software Process Models and Activities

- **Software Process**
  - Definition
    - Software Development Process — A set of related activities that leads to the production of a software product.
  - Fundamental Activities
    - Software Specification (Requirement Engineering/Studies)
      - Aims to produce an agreed requirements document that specifies a system satisfying stakeholder requirements
      - Requirements
        - High-level statements for end-users and customers
        - Detailed system specification for system developers
      - Main Activities (ref diagram)
        - Feasibility Study
        - Requirements Elicitation and Analysis
          - Collect and understand user requirements
        - Requirements Specification
        - Requirements Validation
          - Test and check to ensure requirements are correct
    - Software Design and Implementation (Software Development)
      - Converting a system specification into an executable system
      - Designing the structure/architecture of the software, data models and strcutures (ER Diagram), interfaces
      - Stages (ref diagram)
      - Design Activities
        - Architectural Design
        - Interface Design
        - Component Design
        - Database Design
      - Design Output
        - Code for agile methods
        - Seperate specification documentation for waterfall models
    - Software Validation (Software Testing)
      - Verification and Validation (V & V)
      - Program testing
        - Development testing
        - System testing
        - Acceptance testing
      - Validation
        - Inspection & Review
          - Check the specification and design
          - Backdoor: A way to access the software without authentication
        - Starts in every steps of software process
      - Stages (ref diagram)
    - Software Evolution

        - Software engineering is an evoltionary process
    - Supporting Activities
        - Documentation
        - Software Configuration Management
        - Risk Management
- **Software Process Types**
    - Approach
        - Sequential Approach
        - Iterative Approach
    - System
        - Critical System
            - Structured
        - Business System
            - Less formal, flexible
    - Process
        - Plan-driven Process
            - All process activities are planned in advance
        - Agile Process
            - Planning and development is incremental and iterative — easier to change according to customer requirements
- **Waterfall Model**
    - Plan-driven Process
        - Plan first
    - Stages (ref to the diagram)
        - Requirements Analysis and Definition
            - WHAT the users want
        - System and Software Design
            - HOW to achieve it
        - Implementation and Unit Testing
            - Coding and Testing
        - Integration and System Testing
            - Combining and Testing
        - Operation and Maintenance
            - Pushing into Production Environment
    - Visibility
        - Documentation is produced at each phase
    - Advantages
        - Suitable for projects that require formal documentation (Contracts need documentations)
    - Disadvantages
        - Inflexibility
            - Inflexible partitioning of the project into distinct stages
        - Difficult Resonse
            - Difficult to respond to changing requirements
        - High Cost
            - Iterations can be costly and involve significant rework

    - Limitation
        - Should only be used when the requirements are well understood and unlikely to change radically during system development
- **Incremental Development**
    - Agile Process
        - Begin by most important features, leaving less important features in later iterations
    - Stages (ref to the diagram)
        - Outline Description
        - Concurrent Activities
            - Specification
            - Development
            - Validation
        - Versions: Initial, Intermediate, Final
    - Advantages
        - Reduced Cost
            - Cost of accommodating changing customer requirement is reduced (not affecting previous work)
        - Accurate Feedback
            - Easier to get customer feedback (being continuously used)
        - Rapid Delivery and Deployment
            - More ... to the customer, even if the functions are not completely included.
    - Disadvantages
        - Lack Visibility
            - Produce very few documentations
                - Maybe required by external regulations (accounting system/realtime system)
                - Difficult to create a contract
        - System Stucture Degradation
            - Changes very often => messy code => affect the overall structure
        - Mismatch (Unstable Framework/Architecture)
            - Hard to match between the procedures after a long time (no documentation), especially for large, complex and long-lifetime systems
- **Reuse-oriented Software Engineering**
    - Reusing existing software (components)
        - Rely on a large base of reusable software components and an integrating framework for the composition of these components
    - Stages (ref to the diagram)
        - Requirements Specification
        - Component Analysis
        - Requirement Modification
        - System Design with Reuse
        - Development and Integration
        - System Validation
    - Software Components Types
        - Web Services
        - Collection of Objects
            - Integrated with a component framework like .NET or J2EE

        - Stand-alone Software Systems
    - Advantages
        - Reduced Costs and Risks
            - Reducing the amount of software to be develop
        - Faster Delivery
    - Disadvantages
        - Inevitable Requirement Compromises
        - Control Lost
            - Some control over the system evolution is lost as new versions of the reusable component