CHAPTER 2: SOFTWARE PROCESS
Prayer

- Allahumma Ya Alim Ya Hakim Ya Rashid, kurniakanlah kami dengan ilmu, kepintaran dan kebijaksanaan kepada kami supaya kami dapat menggunakan ilmuMu ke arah kebaikan.

- Allahumma Ya Ba’its, kurniakan kami dengan semangat untuk menuntut ilmuMu
SOFTWARE PROCESS

- Is a **set of activities** and associated results which lead to production of a software product.
- Involve the development of software from **scratch** although the software is developed by extending and modifying existing system.
- There is no ideal process and different organisations have developed completely different approaches to software development.
SOFTWARE PROCESS ACTIVITIES
SOFTWARE PROCESS ACTIVITIES

- A structured set of activities required to develop a software system
  - Specification;
  - Design;
  - Validation;
  - Evolution.

- Software Development Life Cycle (SDLC)
1. Software Specifications

- The process of *establishing what services are required* and the *constraints* on the system’s operation and development
- **Requirements engineering process** is software specification development
  - to produce an *agreed requirements document* that specifies a system satisfying stakeholder requirement
Requirement Engineering

1. Feasibility Study
   - Feasibility Report

2. Requirement Elicitation and Analysis
   - System Models

3. Requirement Specifications
   - User & System Requirements

4. Validation
   - Requirement Documents

Feasibility Report
System Models
User & System Requirements
Requirement Documents
2. Software Design & Implementations

- The process of converting the system specification into an executable system
- Software design: design a software structure that realizes the specification
- Implementation: translate this structure into an executable program
- The activities of design and implementation are closely related and may be inter-leaved
General Model of Design Models

Design Inputs
- Platform Information
- Requirement Specification
- Data Description

Design Activities
- Architecture Design
- Interface Design
- Component Design
- Database Design

Design Outputs
- System Architecture
- Database Specifications
- Interface Specifications
- Component Specifications
Design Activities

- **Architecture Design**: identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships and how they are distributed.

- **Interface Design**: define the interfaces between system components.

- **Component Design**: design how each system component will operate.

- **Database Design**: design the system data structures and how these are to be represented in a database.
3. Software Validations

- Verification and validation (V & V) is intended to show that a system confirms to its specification and meets the requirements of the customer.
- V & V involves checking and review processes and system testing.
- System testing involves executing the system with test cases taken from the specification of the real data.
## Testing Stages

- **Component Testing**
  - Individual components are tested independently
  - Components maybe functions or objects or coherent groupings of these entities

- **System Testing**
  - Testing of the system as a whole
  - Testing properties is particularly important

- **Acceptance Testing**
  - Testing with customer data to check that the system meets the customer’s needs
Testing in Plan-Driven Software Process
4. Software Evolution

- Software is inherently flexible and can change.
- The software evolve and change due to changing requirement of changing business circumstances.
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.
Software Evolution

1. Define System Requirement
2. Access Existing Systems
3. Propose System Changes
4. Modify Systems

Existing System
New System
THE END OF PART 1
THANK YOU
GENERAL SOFTWARE PROCESS MODELS
General Software Process Models

- Structured set of activities required to develop a software system
- Many different software processes but all involve:

<table>
<thead>
<tr>
<th>PROCESS</th>
<th>TASK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Specification</td>
<td>defining what the system should do</td>
</tr>
<tr>
<td>2. Development (Design and Implementation)</td>
<td>defining the organization of the system and implementing the system</td>
</tr>
<tr>
<td>3. Validation</td>
<td>checking if the software produce meets the requirement by the customer</td>
</tr>
<tr>
<td>4. Evolution</td>
<td>changing the system in response to changing customer needs</td>
</tr>
</tbody>
</table>
Software Process Model

- is an abstract representation of a process
- presents a description of a process from some particular perspective
Software Process Descriptions

- Description and discussion regarding process usually covers activities in the process
  - e.g. specifying a data model, designing a user interface, and the ordering of these activities

- Process descriptions may also include:
  - **Products**: the outcomes of a process activity
  - **Roles**: the responsibilities of the people involved in the process
  - **Pre- and post-conditions**: statements that are true before and after a process activity has been enacted or a product produced
Software process are categorized into 2 types:

- Plan-driven processes
- Agile process

<table>
<thead>
<tr>
<th>Plan-driven Process</th>
<th>Agile Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>all of the process activities are planned in advance, progress is measured against this plan.</td>
<td>planning is incremental (done phase by phase), easier to change the process to comply with changing customer requirements</td>
</tr>
</tbody>
</table>

In practice, most practical processes include elements of both plan-driven and agile approaches

- There are no right or wrong software processes
1. The waterfall model
   - Plan-driven model
   - Separate and distinct phases of specification and development

2. Incremental development
   - Specification, development and validation are interleaved
   - May be plan-driven or agile

3. Reuse-oriented software engineering
   - The system is assembled from existing components
   - May be plan-driven or agile

In practice, most large systems are developed using a process that incorporates elements from all of these models.
WATERFALL MODEL

1. Requirements definition
   - System's services, constraints and goals are established by consultation with system users

2. System and software design
   - Allocates the requirements to hardware and software to produce the overall system architecture

3. Implementation and unit testing
   - The software design is realized as a program unit, and unit testing will verify that each unit meets its specs

4. Integration and system testing
   - Program units are integrated and tested as complete system according to its specs and delivered to customer.

5. Operation and maintenance
   - The system is installed for practical use. Maintenance includes correcting errors, and enhancing the system as per requirements
Phases in the waterfall model:
1. Requirements definition
2. System and software design
3. Implementation and unit testing
4. Integration and system testing
5. Operation and maintenance

Main drawback of the waterfall model:
- the difficulty of accommodating change after the process is underway
- a phase has to be complete before moving onto the next phase.
Problems:

- Inflexible partitioning of the project into distinct stages cause difficulty to respond to changing customer requirements
  - only appropriate when the requirements are well-understood and changes will be fairly limited during the design process

- Mostly used for only large systems engineering projects where a system is developed at several sites
  - the plan-driven nature of the waterfall model helps coordinate the work
**INCREMENTAL DEVELOPMENT**

- A model that based on the idea of developing initial implementation, exposing to user comment and evolving it through several versions until adequate system has been developed

- Specification, development and validation activities are interleaved rather than separate

- Rapid feed back across the activities
INCREMENTAL DEVELOPMENT

Concurrent Activities

Outline Description

Specification → Development ← Validation

→ Initial version ← Intermediate Version ← Final Version
Benefits:

1. The cost accommodating changing customer requirements is reduced
   - analysis and documentation that has to be redone is much less than the waterfall model

2. Easier to get customer feedback on the development work that has been done
   - customers can comment on demonstrations of the software and see how much has been implemented

3. More rapid delivery and deployment of useful software to the customer is possible
   - customers are able to use and gain value from the software earlier than the waterfall process
Problems:

1. 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

2. System structure tends to degrade as new increments are added
   - regular change tends to corrupt the process structure
   - incorporating further software changes becomes increasingly difficult and costly
REUSE-ORIENTED SOFTWARE ENGINEERING

- Been used informally when people are working on the project know of designs or code which is similar to that required
  - They will modify them as required and incorporate them into their system
- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems
- Reuse is now the standard approach for building many types of business system
REUSE-ORIENTED SOFTWARE ENGINEERING

1. Component search based on specification for implementation
2. Requirement Modification
3. System Design with Reuse
4. Development and Integration
5. Requirement Specification

Requirement Specification

- Component Analysis
- Requirement Modification

System Validation

- The software that cannot be procured is developed, and components and COTS system are integrated
- The reused component are considered for framework design. New software has to be added if not available.
Types of software components:

1. Web services which are available for remote invocation.

2. Collections of objects that are developed as a package to be integrated with a component framework, e.g., .NET or J2EE.

3. Stand-alone software systems (COTS) that are configured for use in a particular environment.
CHANGING SYSTEM REQUIREMENTS
Coping with change

- Change is inevitable in all large software projects. **Business changes** lead to new and changed system.
- **New technologies** open up new possibilities for improving implementations.
- Change leads to rework so the costs of change include both rework and implementing new functionality.
- Processes need to be structured for iterative development and delivery so that changes is possible without disrupting the whole system.
- Two ways of coping:
  1. System Prototyping
  2. Incremental Delivery
A prototype is an initial version of a system used to demonstrate concepts and try out design options. A prototype can be used in:

- The requirements engineering process to help with requirements elicitation and validation
- Design processes to explore options and develop a UI design
- Testing process to run back-to-back tests
Benefits:

1. Improved system usability
2. A closer match to users’ real needs
3. Improved design quality
4. Improved maintainability
5. Reduced development effort
Incremental Delivery

- System increments are delivered to the customer for comment and experimentation.
- User requirements are prioritized and the highest priority requirements are included in early increments.
- Once the development of the increment is started, the requirements are frozen though the requirements for later increments can continue to evolve.
**Incremental Delivery**

**Incremental Development**
- Develop the system in increments and evaluate each increment before proceeding to the development of the next increment.
- Normal approach used in agile methods.
- Evaluation done by user/customer proxy.

**Incremental Delivery**
- Deploy an increment for use by end-users.
- More realistic evaluation about practical use of software.
- Difficult to implement for replacement systems as increments have less functionality than the system being replaced.
Advantages Incremental Delivery

- Customer value can be delivered with each increment so system functionality is available earlier
- Early increments act as a prototype to help elicit requirements for later increments
- Lower risk of overall project failure
- The highest priority system services tend to receive the most testing
Problems:

- Most systems require a set of basic facilities that are used by different parts of the system.
- Iterative development can also be difficult when a replacement system is being developed.
- The essence of iterative processes is that the specification is developed in conjunction with the software.
Iterative Software Process Models
Process is represented as a spiral rather than as a sequence of activities with backtracking.

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.
BOEHM’S SPIRAL MODEL

- Each loop in the spiral represents a phase of the software process.
  - Each loop split into four sectors
  1. Objective setting - more on system feasibility
  2. Risk assessment and reduction - detailed analysis
  3. Development and validation
  4. Planning
Boehm’s Spiral Model

- Spiral model has been very influential in helping people think about iteration (cycle repetition) in software processes and introducing the risk-driven approach to development.
THE RATIONAL UNIFIED PROCESS

- An iterative software development process framework created by IBM
- An adaptable process framework, intended to be customized by the development organization and software developer
- The element process are selected according to necessity
- Normally described from 3 perspectives
  - A dynamic perspective that shows phases over time
  - A static perspective that shows process activities
  - A practice perspective that suggests good practice
THE RATIONAL UNIFIED PROCESS

Phase:

1. Inception
   - Establish the business case for the system

2. Elaboration
   - Develop an understanding of the problem domain and the system architecture

3. Construction
   - System design, programming and testing

4. Transition
   - Deploy the system in its operating environment
Iteration:

1. In-phase iteration
   - Each phase is iterative with results developed incrementally

2. Cross-phase iteration
   - As shown by the loop in the RUP model, the whole set of phases may be enacted incrementally
RUP fundamental Practice:

- i. Develop software iteratively
  - Plan increments based on customer priorities and deliver highest priority increments first

- ii. Manage requirements
  - Explicitly document customer requirements and keep track of changes to these requirements

- iii. Use component-based architectures
  - Organize the system architecture as a set of reusable components
iv. Visually model software

Use graphical UML models to present static and dynamic views of the software

v. Verify software quality

Ensure that the software meet’s organizational quality standards

vi. Control changes to software

Manage software changes using a change management system and configuration management tools
END