Requirements and Design

- Requirements Phase
- Design Phase
- Architectural Design
- Detailed Design
A requirement is a user need or a feature of the system or a description of something the system is capable of doing in order to fulfill the purpose of the system.

The requirements may be expressed in two ways:

- **Itemized format** – This is in a conventional verbose form written in simple English (structured natural language). It is also called *Problem Summary Statement*.

- **Visual modeling** – UML is one of the most popular visual modeling language to express the requirements.
Requirements definition and specification

The first stage of Requirements phase is Requirements elicitation and analysis. There are 3 steps:

- Problem analysis – Have we captured all the user need?
- Problem description – Are we using the right techniques or views?
- Prototyping and testing – Is this function feasible?
The second stage of Requirements phase is Requirements definition and specification. There are 2 steps:

- **Documentation** – are all the requirements properly defined and documented?

- **Validation** – are all the requirements verified and validated?
The Requirements should have the following characteristics:

• Correct - Are the requirements correct without error?
• Consistent – Are there any conflicting or ambiguous requirements?
• Complete – Are all states, state changes, inputs, outputs, process, features and constraints of the software described by some requirement?
• Realistic – Can what the customer is asking the system to do really be done?
• Verifiable – Are we able to write tests that demonstrate that the requirements have been met?
Example of Requirements

• Consider a simple case study of ATM of Riyadh bank. In this example a customer can withdraw and deposit money through the ATM services of the Riyadh bank.

• We will produce the requirements for this case study using the itemized format in plain English
Requirements in itemized format

1) Initially a customer opens an account with 500 riyals.
2) The customer tracks the balance in his account through the ATM.
3) There are two buttons on the screen of ATM: Withdraw and Deposit.
4) The customer can withdraw money from his account, which decreases his balance.
5) The customer can deposit money in his account, which increases his balance.
6) Whenever money is withdrawn or deposited, a message is displayed confirming the action.
7) If the customer tries to withdraw money, which is more than the balance, an error message is displayed.
8) Here is a restriction on the ATM transaction of a customer. A customer can withdraw or deposit exactly 50 riyals at a time.
The next step in the SDLC is the Design phase which translates the requirements into the solution.

During the design phase, the designers decide how the software system should satisfy the requirements identified during the Requirements Phase.
The levels of Design Phase

- The Software Design phase has two levels:
  - Architectural Design – This associates the requirements specification with the system components that will implement them. Components are usually modules, and the architecture also describes the interconnections among them.
  
  - Detailed Design – It addresses the code design of each module. This involves algorithms, data structures and procedures.
ATM Example

- Building from the ATM Example at Riyadh Bank
- We take the requirements and identify the classes that we will use to implement these requirements
- Example of an Architectural Design on Next Slide
Architectural Design

- A class diagram showing the classes and their association

<table>
<thead>
<tr>
<th>Customer</th>
<th>Account</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Name: String</td>
<td>- Balance: double</td>
</tr>
<tr>
<td>- ID: String</td>
<td>+ Deposit(double): void</td>
</tr>
<tr>
<td>+ getName(): String</td>
<td>+ Withdraw(double): double</td>
</tr>
<tr>
<td>+ setName(String): void</td>
<td></td>
</tr>
<tr>
<td>+getID(): String</td>
<td></td>
</tr>
</tbody>
</table>
Detailed Design

- Detailed Design involves the actual working of each interface in a module. In a more OO friendly language,
  - Define how each method in a class will be implemented

- A Detailed Design for the methods of Account class is given on the next slide
Detailed Design Example

• Account Class Methods
  – public void Deposit( double amount)
    Add amount to balance and then return
  
    – public double Withdraw (double amount)
      if(amount is less than Balance)
      return -1
      otherwise
      Subtract Amount from Balance
      return new Balance
Characteristics of Good Design

• Component independence - Are the components reusable, portable and expandable in the future?

• Exception Identification and Handling – Is exception handling properly embedded in the design so that the system will be secured and autonomous?

• Fault Prevention and Fault Tolerance – Do designers anticipate faults and handle them in ways that minimize disruption and maximize safety? Do designers guard against faults built into each component, as well as against faults introduced by other components, systems, and interfaces?
Exercise

• What is wrong with the following requirements
  – accuracy shall be sufficient to support mission planning
  – system shall provide timely response to queries
  – The System shall not crash if a process experiences a faulty situation
  – The System shall keep all values up to date