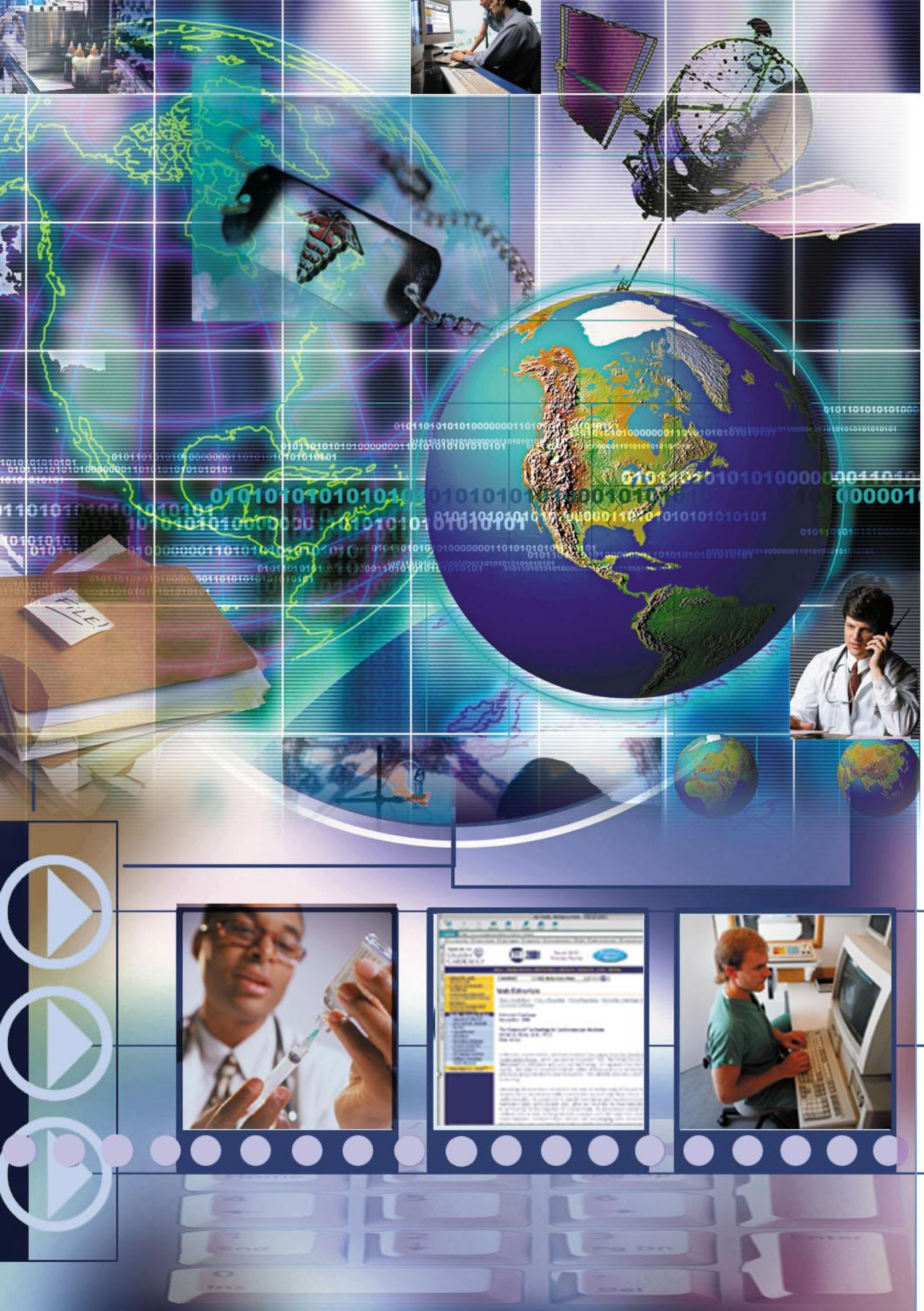


# Chapter 1:
## THE ROLE OF SOFTWARE DESIGN

By:
Ts. Azaliza binti Zainal

# THE NATURE OF DESIGN PROCESS

- **Purpose of Design**: To produce a workable (implementable) solution to a given problem.
- Main Characteristics Found in Almost All Design Problems:
  - No Single "Right" Solution
  - Many Factors and Constraints to be Balanced in Choosing a Solution
  - No One Measure of "Quality"
  - No Particular Process That Can Ensure That We Can Even Identify an Acceptable Solution.

# DESIGN – PROBLEM-SOLVING APPROACH

- **Fitness for Purpose**
  - The Key Measure of the Appropriateness of Any Solution

- Is There a Systematic Approach to Design?

  No, a Designer Must Create Each System
  - Identify the Properties Required:

    Stake Holders (Customer, Users, etc.)
  - Devise a Structure That Possesses the Properties

# SOFTWARE DESIGN PROCESS

- **Designer Formulates and Develops** an Abstract Design Model Representative of the Solution.

- Why is This Process Not Understood as Well as Other Forms of Design?
  - The Complexity of Software
  - The Problem of Conformity
  - The (Apparent) Ease of Changeability
  - The Invisibility of Software

# SOFTWARE DESIGN PROCESS

- Software Design Method:
  - Used When a Designer Lacks Experience or is Unfamiliar With the Problem to be Solved
  - Limited to Forms of Design Practice That Can be Prescribed in a Procedural Manner.
- These Methods Provide:
  - A Representation Part
  - A Process Part
  - A Set of Heuristics

# DESIGN CONSTRAINTS

- Designing Software is Rarely an Unconstrained Process

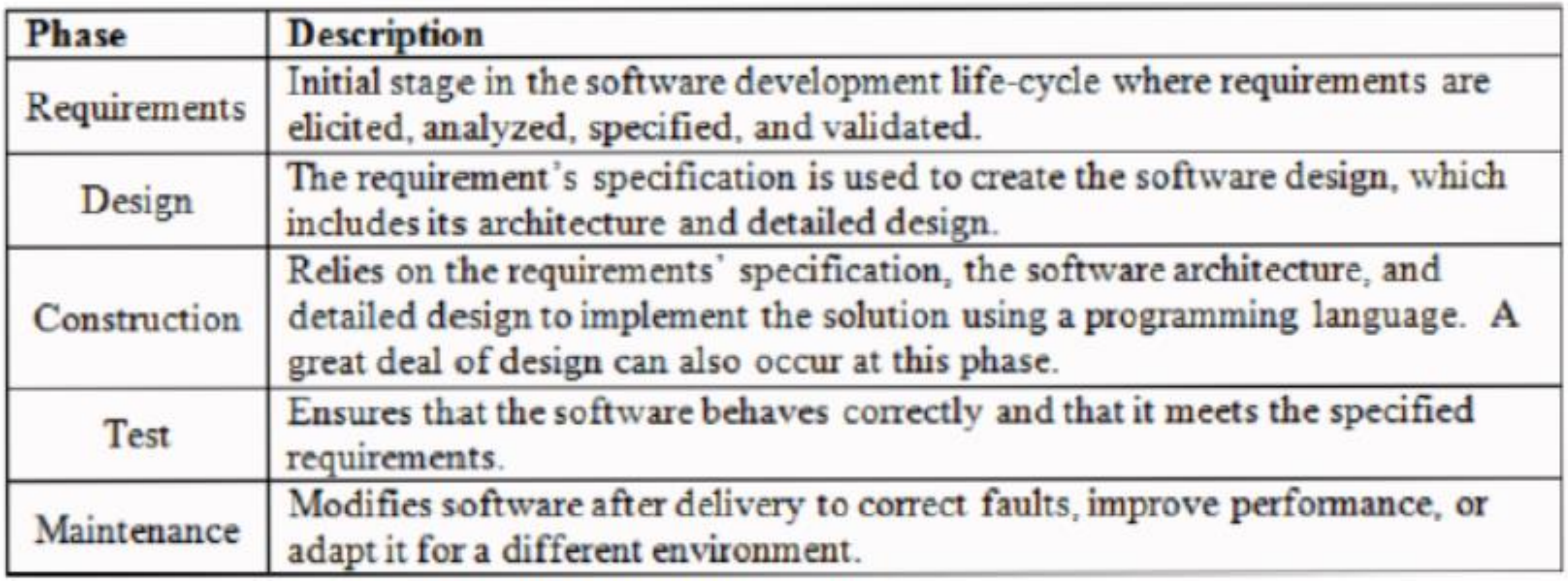
- Examples of Constraints:
  - Programming Language to be Used
  - Execution Environment or Operating System
  - Performance Expectations
  - User Interface Needs

# DESIGN IN SOFT DEVELOPMENT CYCLE

- Constraints Affect the Design Process and the Form of the Product.
- Set of User Needs to be Met
  - **Fitness of Purpose**
  - Requirements Elicitation and Analysis: Leads to Identifying Inconsistencies Between the Requirements and the Solution
- Designer Must “Think Ahead”
  - Short Term Use, Long Maintenance Effort, Stability of the Solution Space, etc.

# DESIGN IN SOFT DEVELOPMENT CYCLE

- A formal process for engineering software includes the following phases:



| Phase | Description |
|---|---|
| Requirements | Initial stage in the software development life-cycle where requirements are elicited, analyzed, specified, and validated. |
| Design | The requirement's specification is used to create the software design, which includes its architecture and detailed design. |
| Construction | Relies on the requirements' specification, the software architecture, and detailed design to implement the solution using a programming language. A great deal of design can also occur at this phase. |
| Test | Ensures that the software behaves correctly and that it meets the specified requirements. |
| Maintenance | Modifies software after delivery to correct faults, improve performance, or adapt it for a different environment. |

- Of importance to this course is the design phase, where requirements are used to create a blueprint of the software to be constructed.

# DESIGN QUALITIES

- **Fitness of Purpose** Doesn’t Provide an Absolute Measure of Quality
  - Correct and Within Constraints May Not be Enough to Achieve Fitness of Purpose
- Quality Factor “ilities”:
  - Reliability
  - Efficiency
  - Maintainability
  - Usability

# ASSESSING DESIGN QUALITY

- A Systematic Form of Measurement is Difficult to Achieve

- Favourable Assessment Techniques
  - Design Walk-through Meetings
  - Reviews
  - Refactoring (XP)

- How Often?

# ENGINEERING DESIGN

- Design is not a new concept conceived by software engineers. Design is used in all other engineering disciplines, e.g., electrical, mechanical, civil, etc. Dym and Little define engineering design as:
  - A systematic, **intelligent process** in which designers generate, evaluate and specify designs for devices, systems or processes whose form(s) and function(s) achieve clients' objectives and users' needs while satisfying a specified set of constraints.

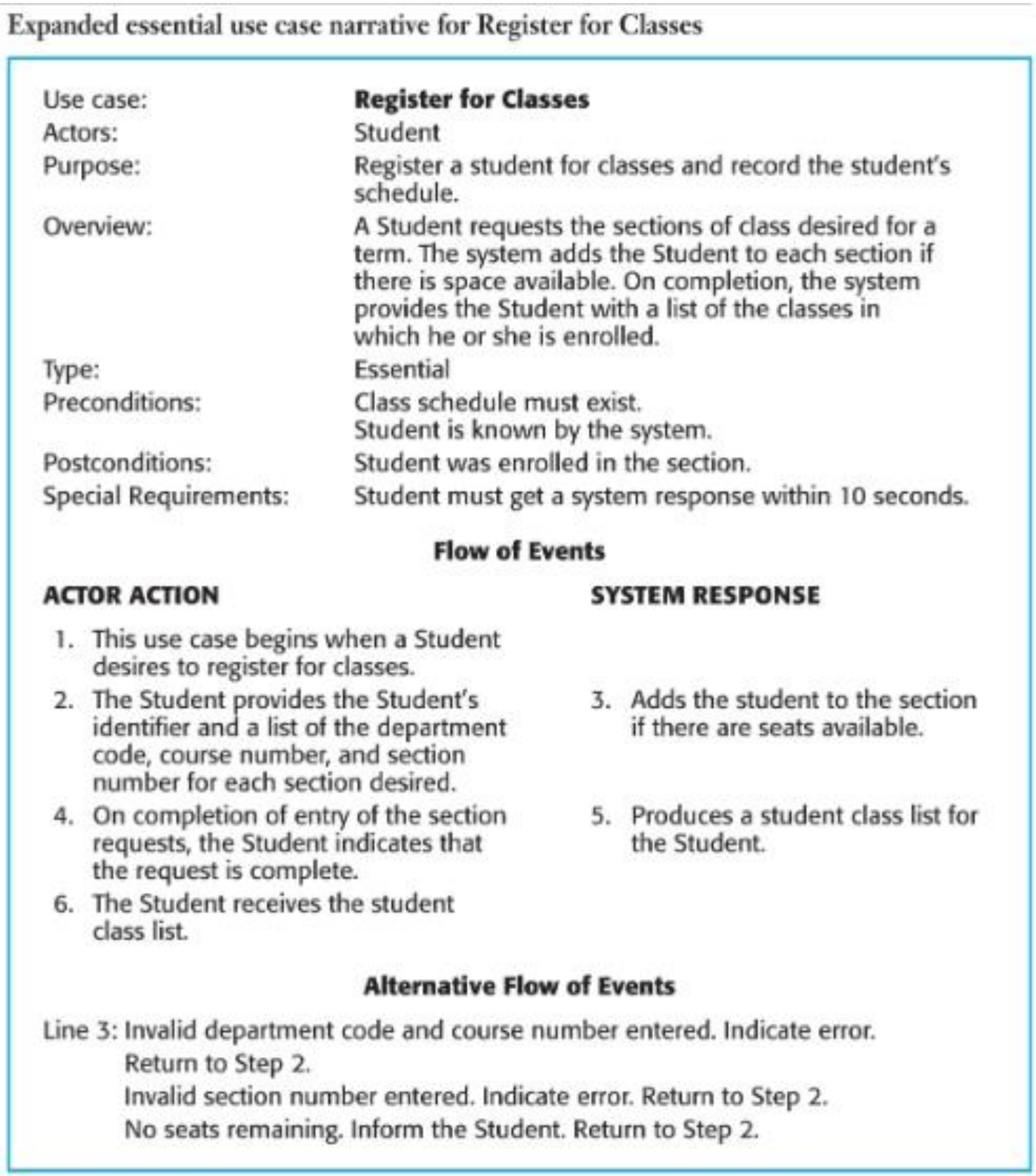
# ENGINEERING PROBLEM SOLVING

- What do we mean by intelligent process?
  - We refer to it as intelligent process because a great deal of problem-solving occurs during design.
  - During the design process, engineers are constantly engaging in problem solving activities.
    - Their work requires them to identify, evaluate, and proposed solutions to complex problems.
    - Some problems encountered by engineers have never been solved before!

# ENGINEERING PROBLEM SOLVING

- Because of the large number of problem-solving activities present during design, a formal discussion on problem-solving is required.
  - To become a good designer, engineers must be good problem solvers.
  - To become a good problem solver…
    well, that requires lots of time and effort,
    but we can at least set up a framework for
    solving problems throughout the design
    phase.

# SUMMARY



Expanded essential use case narrative for Register for Classes

| | |
|---|---|
| Use case: | **Register for Classes** |
| Actors: | Student |
| Purpose: | Register a student for classes and record the student's schedule. |
| Overview: | A Student requests the sections of class desired for a term. The system adds the Student to each section if there is space available. On completion, the system provides the Student with a list of the classes in which he or she is enrolled. |
| Type: | Essential |
| Preconditions: | Class schedule must exist.<br>Student is known by the system. |
| Postconditions: | Student was enrolled in the section. |
| Special Requirements: | Student must get a system response within 10 seconds. |

**Flow of Events**

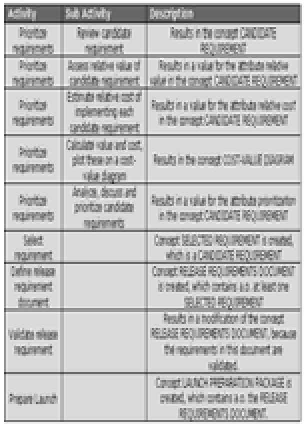
| **ACTOR ACTION** | **SYSTEM RESPONSE** |
|---|---|
| 1. This use case begins when a Student desires to register for classes. | |
| 2. The Student provides the Student's identifier and a list of the department code, course number, and section number for each section desired. | 3. Adds the student to the section if there are seats available. |
| 4. On completion of entry of the section requests, the Student indicates that the request is complete. | 5. Produces a student class list for the Student. |
| 6. The Student receives the student class list. | |

**Alternative Flow of Events**

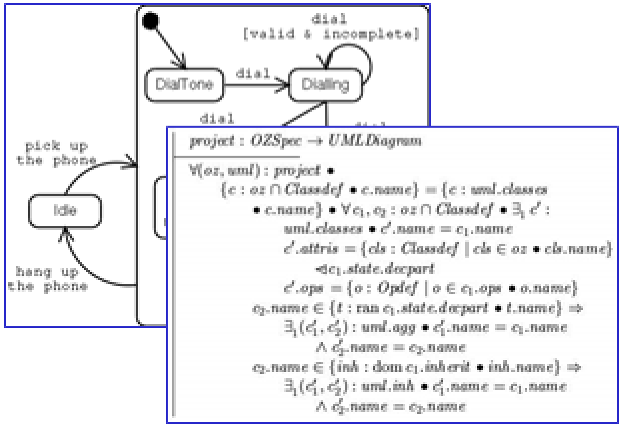
Line 3: Invalid department code and course number entered. Indicate error.
Return to Step 2.
Invalid section number entered. Indicate error. Return to Step 2.
No seats remaining. Inform the Student. Return to Step 2.

# SUMMARY

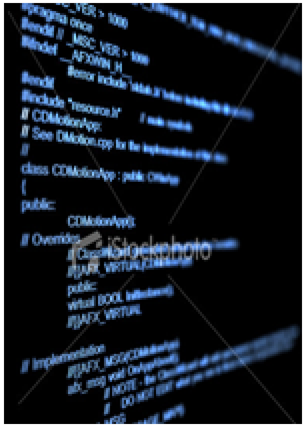
## The Role of Design



| Activity | Sub Activity | Description |
|---|---|---|
| Prioritize requirements | Review candidate requirement | Results in the concept CANDIDATE REQUIREMENT |
| Prioritize requirements | Assess relative value of candidate requirement | Results in a value for the attribute relative value in the concept CANDIDATE REQUIREMENT |
| Prioritize requirements | Estimate relative cost of implementing each candidate requirement | Results in a value for the attribute relative cost in the concept CANDIDATE REQUIREMENT |
| Prioritize requirements | Calculate value and cost, plot these on a cost-value diagram | Results in the concept COST-VALUE DIAGRAM |
| Prioritize requirements | Analyze, discuss and prioritize candidate requirements | Results in a value for the attribute prioritization in the concept CANDIDATE REQUIREMENT |
| Select requirement | | Concept SELECTED REQUIREMENT is created, which is a CANDIDATE REQUIREMENT |
| Define release requirement document | | Concept RELEASE REQUIREMENTS DOCUMENT is created, which contains a.o. at least one SELECTED REQUIREMENT |
| Validate release requirement | | Results in a modification of the concept RELEASE REQUIREMENTS DOCUMENT, because the requirements in this document are validated. |
| Prepare Launch | | Concept LAUNCH PREPARATION PACKAGE is created, which contains a.o. the RELEASE REQUIREMENTS DOCUMENT. |



$$
\begin{array}{l}
project : OZSpec \rightarrow UMLDiagram \\
\hline
\forall (oz, uml) : project \bullet \\
\quad \{c : oz \cap Classdef \bullet c.name\} = \{c : uml.classes \\
\qquad \bullet\ c.name\} \bullet \forall c_1, c_2 : oz \cap Classdef \bullet \exists_1 c' : \\
\qquad\quad uml.classes \bullet c'.name = c_1.name \\
\qquad\quad c'.attris = \{cls : Classdef \mid cls \in oz \bullet cls.name\} \\
\qquad\qquad \lhd c_1.state.decpart \\
\qquad\quad c'.ops = \{o : Opdef \mid o \in c_1.ops \bullet o.name\} \\
\qquad c_2.name \in \{t : \operatorname{ran} c_1.state.decpart \bullet t.name\} \Rightarrow \\
\qquad\quad \exists_1 (c'_1, c'_2) : uml.agg \bullet c'_1.name = c_1.name \\
\qquad\qquad \wedge\ c'_2.name = c_2.name \\
\qquad c_2.name \in \{inh : \operatorname{dom} c_1.inherit \bullet inh.name\} \Rightarrow \\
\qquad\quad \exists_1 (c'_1, c'_2) : uml.inh \bullet c'_1.name = c_1.name \\
\qquad\qquad \wedge\ c'_2.name = c_2.name
\end{array}
$$



```
[illegible]_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#ifndef __AFXWIN_H__
    #error include [illegible]
#endif
#include "resource.h"        // [illegible]
// CDMotionApp:
// See DMotion.cpp for the [illegible]
//
class CDMotionApp : public CWin[illegible]
{
public:
    CDMotionApp();
// Overrides
    // Class[illegible]
    //{{AFX_VIRTUAL(CDMotion[illegible]
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL
// Implementation
    //{{AFX_MSG(CDMotionApp)
    afx_msg void OnAppAbout();
        // NOTE - the Cl[illegible]
        //    DO NOT EDIT [illegible]
    //}}[illegible]
```

Requirements

Design Models

Code

Design should play a pivotal role:

- *Clarify and refine requirements*
- *Early defect detection and elimination*

# SUMMARY

- Design was introduced as a systematic and intelligent process for generating, evaluating, and specifying designs for devices, systems, or processes.

- In software engineering, design provides blueprints that capture how software systems will meet their required functions and how they will be shaped to meet their intended quality.