2209 SE351/CS480-06 Course Outline

Subject Code	:	**SE351/CS480-06**
Subject Title	:	SOFTWARE DESIGN
Course Type	:	Compulsory/ /Elective
Level	:	4
Credits	:	3
Teaching Activity	:	Lecture 45 hours
Prior Knowledge*	:	(CS 110 / SE 110) Object-oriented programming(*)
Class Schedule	:	

Class	Week	Time	Classroom	Date
D1	THU	12:30-15:20	C508	2023/09/04-2023/12/17
D1	THU	12:30-15:20	C508	2023/09/04-2023/12/17

Instructor	:	Ye Ben
Contact Number	:	8897-3022
E-mail Address	:	bye@must.edu.mo
Office	:	A307b
Office Hour	:	

MON	10:30—12:30
TUE	9:30—12:30
WED	9:30—12:30
THUR	15:30—17:30

COURSE DESCRIPTION

The course aims to introduce the principle of design pattern and present program design guidelines that show the students how to design an elements of reusable software program. In software engineering, a design pattern is a general repeatable solution to a commonly occurring problem in software design. A design pattern isn't a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.

Design patterns can speed up the development process by providing tested, proven development paradigms. Effective software design requires considering issues that may not become visible until later in the implementation. Reusing design patterns helps to prevent subtle issues that can cause major problems and improves code readability for coders and architects familiar with the patterns.

Often, people only understand how to apply certain software design techniques to certain problems. These techniques are difficult to apply to a broader range of problems. Design patterns provide general solutions, documented in a format that doesn't require specifics tied to a particular problem.

In addition, patterns allow developers to communicate using well-known, well understood names for software interactions. Common design patterns can be improved over time, making them more robust than ad-hoc designs.

TEXT BOOK

E.Gamma, R.Helm, R.Helm, and John Vlissides, Design Pattern – Elements of Reusable Object-oriented Software. 1994

REFERENCE

1) E. Freeman, and E. Robson, Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software, 2nd Edition, O'Reilly Media, 2021
2) K. Ayeva, and S. Kasampalis Mastering Python Design Patterns: A guide to creating smart, efficient, and reusable software, 2nd Edition, Packt Publishing, 2018

INTENDED LEARNING OUTCOMES

Upon successful completion of this subject, students will be able to:

1. Understand the concepts of software design and design principles
2. Understand the importance of software design at the complete process of software engineering
3. Understand the difference design patterns and their influence for software projects
4. Ability to use design patterns to build good software architectures.
5. Understand how to design good software architecture and learn the relationship between software architecture and project cost.

Weekly Schedule

Week	Topic	Hours	Teaching Method
1	Overview of Software Process	3	lecture
2	Software Design and Software Process	3	lecture
3	Software Architecture	3	lecture
4	Relationship between Software Design and Software Architecture	3	lecture
5	Design Principles	3	lecture
6	Design Patterns: Part 1	3	lecture
7	Design Patterns: Part 2	3	lecture
8	Program Data	3	lecture
9	Abstracting Designs	3	lecture
10	Designing Algorithms	3	lecture
11	The Cost of Computing and Vectors	3	lecture
12	Designing Functions that Change	3	lecture

	Structures		
13	Paper Presentation & Discussion	3	lecture
14	Paper Presentation & Discussion	3	lecture
15	Review	3	lecture
16	Final Exam Week	2	Exam

ASSESSMENT APPROACH

Assessment method	% weight
1.Attendance (Class participation)	10%
2.Assignment	10%
3.Midterm Exam	30%
4.Final exam	50%
Total	100 %

Guideline for Letter Grade:

Marks	Grade
93-100	A+
88-92	A
83-87	A-
78-82	B+
72-77	B
68-71	B-
63-67	C+
58-62	C
53-57	C-
50-52	D
0-49	F
Marks	Grade

Notes:

Students will be assessed on several assessment items (i.e. attendance, assignments, paper presentation and the final exam.).

The attendance evaluates the student's participation of discussion in the classes.

The final exam evaluates the student's understanding of the concepts of software

engineering.
The paper presentation is used to evaluate the student’s ability to understand the cutting-edge knowledge of software design.