National University of Sciences & Technology (NUST)
School of Electrical Engineering and Computer Science (SEECS)
Department of Software Engineering

Software Design and Architecture

Course Code:	SE-210	**Semester:**	5th
Credit Hours:	3+1	**Prerequisite Codes:**	SE
Instructor:	Dr. Qaiser Riaz	**Class:**	BESE 5AB
Office:	A-103, IAEC Building	**Telephone:**	9085 2184
Lecture Days:	Mon., Thur., and Fri.	**E-mail:**	qaiser.riaz@seecs.edu.pk
Class Room:	TBA	**Consulting Hours:**	Tuesday 2-3 pm (by prior email)
Lab Engineer:	Nadeem Nawaz	**Lab Engineer Email:**	nadeem.nawaz@seecs.edu.pk
Knowledge Group:	Software Engineering	**Updates on LMS:**	After every lecture

Course Description:

Software Design & Architecture course is designed to help students to develop skills that will enable them to construct software of high quality, software that is reliable, easy to understand, modify and maintain. The course is important in relation to the core modules and helps the students to understand the Object oriented software design and to develop reusable software.

The course introduces principles of good design, object oriented software design and a number of architectural patterns like client server, multi layered architecture, Transaction processing, Pipes and filter, service oriented architecture, and cloud computing etc. These architectural patterns help the students to analyze the non-functional requirements. A particular emphasis is laid on the design process so that students can recognize the significance of design in maintaining the project.

Course Objective:

To enhance the abilities of students to develop reusable software designs. To introduce the students to the principles of good design, design approaches, paradigms and object-oriented concepts. In addition to this students will learn software architecture that represents the gross-level structure of software intensive systems and includes early design decisions that impact the quality of the overall system. Software architecture is generally considered to play a fundamental role in coping with the inherent difficulties of the development of large-scale and complex software systems. A common assumption is that architecture design can support the required software system qualities such as robustness, adaptability, reusability and maintainability. This course enables the students to apply architecture pattern, design principles on different real world problems.

Based on students' feedback, special emphasis will be given to the behavioral design patterns and the architectural views and patterns by including a case study on non-functional requirements and choice of architectural patterns. This will help students broadening their knowledge of handling and assessment of non-functional requirements in a real world problem.

Course Learning Outcomes (CLOs):

	Upon completion of the course, students should demonstrate the ability to:		PLO Mapping	BT Level*
	CLO 1	Comprehend software design principles and fundamentals of design/architectural patterns.	PLO 1	C-2
	CLO 2	Select appropriate design pattern and architectural pattern for a given problem.	PLO 2	C-6
	CLO 3	Implement the design pattern and architectural pattern to fulfill given quality criteria	PLO 3	C-5
	CLO 4	Analyze the quality of the implemented design and system architecture	PLO 4	C-6
		* BT= Bloom's Taxonomy, C=Cognitive domain, P=Psychomotor domain, A= Affective domain		

Program Learning Outcomes (PLOs):

PLO 1: Engineering Knowledge
An ability to apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PLO 2: Problem Analysis
An ability to identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

PLO 3: Design/Development of Solutions
An ability to design solutions for complex engineering problems and design systems, components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations.

PLO 4: Investigation
An ability to investigate complex engineering problems in a methodical way including literature survey, design and conduct of experiments, analysis and interpretation of experimental data, and synthesis of information to derive valid conclusions.

PLO 5: Modern Tool Usage
An ability to create, select and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modelling, to complex engineering activities, with an understanding of the limitations.

PLO 6: The Engineer and Society
An ability to apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice and solution to complex engineering problems

PLO 7: Environment and Sustainability
An ability to understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.

PLO 8: Ethics
Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

PLO 9: Individual and Team Work
An ability to work effectively, as an individual or in a team, on multifaceted and /or multidisciplinary settings.

PLO 10: Communication
An ability to communicate effectively, orally as well as in writing, on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PLO 11: Project Management
An ability to demonstrate management skills and apply engineering principles to one's own work, as a member and/or leader in a team, to manage projects in a multidisciplinary environment.

PLO 12: Lifelong Learning
An ability to recognize importance of, and pursue lifelong learning in the broader context of innovation and technological developments.

Topics to be Covered:

1.	Introduction to Software Design and Architecture	2.	Software Design Process and Principles (Class Level)
3.	Software Design Process and Principles (Package Level)	4.	UML Revision and Software Design Patterns
5.	Creational Design Patterns	6.	Structural Design Patterns
7.	Behavioral Design Patterns	8.	Software Architecture I
9.	Architectural Patterns and Styles	10.	Software Architecture Documentation and Evaluation

11. Advanced Topics

Lecture Breakdown:

Week No.	Topics	Assessment	Remarks
1	Introduction to Software Design & Architecture • Importance of software design & architecture • Levels of Designs • Design Approaches • Design Challenges • Characteristics of Good and Bad design • How to know and avoid a bad design		

2	Software Design Process and Principles I
	• Software Design Process
	• Class Level Software Design Principles (SOLID)
	  ○ Single responsibility principle (SRP)
	  ○ The Open-Closed Principle (OCP)
	  ○ The Liskov Substitution Principle (LSP)
	  ○ The Interface Segregation Principle (ISP)
	  ○ The Dependency Inversion Principle (DIP)		

3	Software Design Process and Principles II
	• Structural vs Object-oriented Design
	• Package Level Software Design Principles
	  ○ The Release Reuse Equivalency Principle
	  ○ The Common Closure Principle
	  ○ The Common Reuse Principle
	  ○ The Acyclic Dependencies Principle
	  ○ The Stable Dependencies Principle
	  ○ The Stable Abstractions Principle		

4	UML Revision and Software Design patterns • Design Patterns and their Usage • Classification of Design Patterns (Creational, Structural, Behavioral) • Creational Design Patterns I • Factory Method • Abstract Factory		
5	Creational Design Patterns II • Prototype Pattern • Singleton Pattern • Builder Pattern		
6	**OHT-1**		
7	Structural Design Patterns I • Adaptor Pattern • Decorator pattern • Façade Pattern • Flyweight Pattern		
8	Structural Design Patterns II • Adapter Pattern • Bridge Pattern • Composite Pattern		
9	Behavioral Design Patterns I • Observer Pattern • Interpreter Pattern • State Pattern		

10	Behavioral Design Patterns II • Chain of Responsibility Pattern • Command Pattern • Strategy Pattern
11	Software Architecture • Architectural descriptions using UML • Architecture Description Languages • Architectural views (Logical Process Physical and Deployment views)
12	**OHT-2**
13	Architectural Patterns and Styles I • Architecture Pattern vs Architecture Styles • Layered architectures • Client-Server architectures • Model view controller
14	Architectural Patterns and Styles II • Pipes and filter architecture • Quality and Reusability in Design and Architecture • Case Study – Pattern-based Architecting
15	Software Architecture Documentation and Evaluation • Documenting Software Architecture and Architectural Views • Software Architecture Analysis Method (SAAM) • Architecture Tradeoff Analysis Method (ATAM)
16	Advanced Topics • Service-Oriented Architecture, Design and Architecture for Systems of Systems, IOT, Cyber-physical, Cloud, Mobile-Cloud
17	Project Presentations, demo and viva
18	**ESE**

Lab Experiments:	
Lab 01	Introduction to Rational Rose, Class and Sequence Diagrams
Lab 02	Class, Sequence Diagrams , Activity, State Transition Diagrams
Lab 03	OO Design Principles (Class level)
Lab 04	OO Design Principles (Package level)
Lab 05	Architecture using UML: Package, Deployment, Component Diagrams
Lab 06	Creational Design Patterns
Lab 07	Structural Design Patterns
Lab 08	Behavioral Design Patterns
Lab 09	Architectural Views
Lab 10	Development of Components in the form of Dynamic Link Libraries (DLL)
Lab 11	Implementation of Key Word in Context (KWIC) system using pipes
Lab 12	Implementation of Web based Client Server Application
Lab 13	Model View Controller using ASP.net/Java
Lab 14	Architecture Description Language (ADL)

Tools / Software Requirement:

Rational Rose, Eclipse, J2EE, Visual Studio, ASP.Net

Books:

Text Book:	1. Len Base, Paul Clements, Rick Kazman. Software Architecture in Practice. 3rd Edition, 2012, Addison, Wesley 2. Design Patterns: Elements of Reusable Object-Oriented Software, Erich Gamma, Richard Helm , Ralph Johnson, John Vlissides , Grady Booch
Reference Books:	1. Ian Gortan. Essentials Software Architecture. 2006. Springer. 2. Timothy C. Lethbridge and Robert Laganière. Object-Oriented Software Engineering: Practical Software Development using UML and Java, 2nd Edition. 2005. McGraw Hill. 3. Mary Shaw and David Garlan. Software Architecture - Perspectives on an Emerging Discipline. 1996. Prentice Hall 4. Object-Oriented Systems Analysis and Design using UML, by Simon Bennett, McRobb & Farmer. 2010. McGraw-Hill 5. Evaluating Software Architecture: Methods and Case Studies, by Clements, P.,Kazman, R., and Klein, M. SEI Series in Software Engineering, 2002, Addison-Wesley.

Course Assessment	
Exam:	2 One Hour Tests (OHT) and 1 End Semester Exam (ESE)
Home work:	3 Assignments
Lab Assignments:	12 Reports
Design reports:	1 Design report for the term/semester project
Quizzes:	4 - 5 Quizzes

Course Assessment Weightages (In accordance with NUST statutes)
Theory: 75%
• Quizzes: 15%
• Assignments: 10%
• OHT-1: 15%
• OHT-2: 15%
• End Semester Exam: 45%
Practical: 25%
• Labs Assignments: 70%
• Semester Project: 30%

Grading Policy:	
Quiz Policy:	The quizzes will be unannounced and normally last for ten minutes. The question framed is to test the concepts involved in last few lectures. Number of quizzes that will be used for evaluation is at the instructor's discretion.
Assignment Policy:	In order to develop comprehensive understanding of the subject, assignments will be given. Late assignments will not be accepted / graded. All assignments will count towards the total (No 'best-of' policy). The students are advised to do the assignment themselves. Copying of assignments is highly discouraged and violations will be dealt with severely by referring any occurrences to the disciplinary committee. The questions in the assignment are meant to be challenging to give students confidence and extensive knowledge about the subject matter and enable them to prepare for the exams.
Lab Conduct:	The labs will be conducted for three hours every week. A lab handout will be given in advance for study and analysis The lab handouts will also be placed on LMS. The students are to submit their results by giving a lab report at the end of lab for evaluation. One lab report per group will be required. However, students will also be evaluated by oral viva during the lab.

Semester Project:	The basic aim of the semester project is to help students to practically deal and solve complex engineering problems. Students will propose features of a system having properties of a large-scale complex system, engineer its architectural design, and will implement at least one design pattern of choice from the following: creational, structural, behavioral, architectural design patterns. Each group will have to submit three reports: 1) features of the selected system, 2) architectural design of the selected system, 3) Implementation of selected design patterns. The evaluation will be based on the submitted reports, presentation/demo, and viva.
Plagiarism:	SEECS maintains a zero tolerance policy towards plagiarism. While collaboration in this course is highly encouraged, you must ensure that you do not claim other people's work/ ideas as your own. Plagiarism occurs when the words, ideas, assertions, theories, figures, images, programming codes of others are presented as your own work. You must cite and acknowledge all sources of information in your assignments. Failing to comply with the SEECS plagiarism policy will lead to strict penalties including zero marks in assignments and referral to the academic coordination office for disciplinary action.