University of Science, VNU-HCM
Faculty of Information Technology

# Course Introduction

Assoc. Prof. TRAN Minh Triet
Department of Software Engineering

TRUONG DAI HOC KHOA HOC TU NHIEN
KHTN
TP. HO CHI MINH

## Software Analysis and Design

# Course Objectives

- This course aims to equip students with basic skills to analyze and design software.
- Upon completion this course, students can:
  - Describe the common principles to analyze and design software from software requirements
  - Apply object oriented methods and techniques to analyze and design software
  - Recognize, analyze and evaluate basic pros and cons of an existing analysis or design model, the architecture of a software system, the communication between components in a given system.
  - Apply basic object oriented techniques to optimize analysis/design models to enhance the evolution and flexibility of software systems.

# Summary of Content

- This course introduces the common principles to analyze and design software from software requirements.
- The content of this course focuses on object oriented techniques (using UML) to analyze, to design architecture, interface, business logic, and data.
- Several advanced topics can be optionally introduced (e.g. design patterns, service oriented architecture…)

# Assignments, Projects, and Examinations

- Midterm exam: 2.5 marks
- Final exam: 3.5 marks
- Assignments: 1.0 mark
  - N personal assignments ($5 \leq N \leq 10$)
  - Each assignment 1/N mark
  - Total 1.0 mark

- Final Project 3.0+ marks
  - 2 student(s) per group
  - Documentation: 1.5+ marks
  - Application: 1.5+ marks
- Advanced Topics +1.0 mark
  - TBA

# Content Outline

**Part 1: Overview and Revision**

- Concepts in Software Engineering
  - Software
  - Software Quality
  - Software Engineering
  - Software Processes
  - Object-oriented Software Engineering
- Best Practices in Software Development
  - Develop Iteratively
  - Manage Requirements
  - Use Component Architectures
  - Model Visually (UML)
  - Continuously Verify Quality
  - Manage Change

# Content Outline

**Part 1: Overview and Revision (cont'd)**

- Concepts in Object-Oriented Programming
  - Classes and Objects:
    - Attributes and Methods
    - Visibility: Public, Protected, Private
    - Inheritance, Polymorphism
  - Relationships:
    - Generalization
    - Association, Aggregation, Composition
    - Dependency
  - Other notations

# Content Outline

**Part 2: Requirement Modeling**

- Requirement Modeling with UML – Use-case Diagrams
  - Problem Statement
  - Actors and Use-cases
  - Use-case Model
  - Use-case Diagram
  - Relationships in Use-case Diagrams: generalization, <<include>>, <<extend>>
  - Use-case Specification: Scenarios, Flows of Events, Alternatives, Pre-conditions, Post-conditions…
  - Glossary and Supplemental Specification
  - Examples
- Requirement Modeling with UML – Activity Diagrams

# Content Outline

**Part 3: Use-case Analysis – Static Analysis**

- (Analysis) Class Diagrams
  - Revisions: basic concepts in OOP (again…) and notations in UML (c.f. Part 1)
  - How to enhance the capability for software evolution? (tips and tricks)
  - Lots of examples!

- State Machine diagrams
  - Concepts and notations
  - Examples

# Content Outline

**Part 4: Use-case Analysis – Dynamic Analysis**

- Analysis Classes
- Sequence Diagrams, Communication Diagrams, and VOPCs
  - Concepts and notations
  - Examples

# Content Outline

**Part 5: Data(base) Design**

- Relational Database Design
  - Mapping from a class diagram to a relational database
  - How to enhance the capability for software evolution? (tips and tricks)
  - Lots of examples!

- XML and Semi-structured Data
  - Introduction to XML
  - How to store data using XML
  - Comparison between relational databases and XML-based data

# Content Outline

**Part 6: Software Architecture**

- Introduction to Software Architecture
- Layers and Tiers
- Some guidelines for Software Architecture Analysis and Design

# Content Outline

**Part 7: (User) Interface Design**

- Introduction
  - Layout and behavior of a (user) interface
  - Some common approaches for designing user interfaces
  - Some (common and easy-to-understand) notations
- Examples and applications
  - Data Input Forms: simple object, complex object, relation
  - Search Forms
  - Processing Business Forms
  - Reports
- Several Techniques to Enhance Qualities of User Interfaces
  - Supplemental Information
  - Supplemental Operations
  - Action Acceleration
  - Exception Handlers

# Content Outline

**Part 8: Miscellaneous**

- Design Patterns
- Software Refactoring
- Late-binding functions
- Web services (SOAP, REST) and Service Oriented Architecture
- Model-Driven Architecture
- Mashups and Widgets
- …

# References


- Roger S Pressman, Roger Pressman (2004), Software Engineering: A Practitioner's Approach, McGraw-Hill Science/Engineering/Math
- Ian Sommerville (2006), Software Engineering (8th Edition) , Addison Wesley
- Mike O'Docherty (2005), Object-Oriented Analysis and Design - Understanding System Development with UML 2.0, John Wiley & Sons
- Bernd Oestereich (2001), Developing software with UML - Object-oriented analysis and design in practice, 2nd Edition, Addison Wesley
- Grady Booch et. al. (2007), Object-oriented Analysis and Design with Applications, 3rd Edition, Addison Wesley
- Craig Larman (2001), Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process, 2nd Edition, Prentice Hall PTR
- Dennis de Champeaux, Douglas Lea, Penelope Faure (1993). Software Engineering - Object-Oriented System Development, Addison Wesley
- Len Bass, Paul Clements, Rick Kazman (2003), Software Architecture in Practice, 2nd edition, Addison Wesley
- Sherif M. Yacoub, Hany H. Ammar (2003), Pattern-Oriented Analysis and Design: Composing Patterns to Design Software Systems, Addison Wesley.
- Alan Shalloway, James R. Trott (2004), Design Patterns Explained – A New Perspective on Object Oriented Design, Addison Wesley