Title (Units): # COMP4007 Software Design, Development, and Testing (3,3,1)

Course Aims: This course is aimed to further develop students' knowledge and skills in software engineering, and to introduce and discuss software design patterns, state-of-the-art techniques and advanced topics in developing reliable software systems. At the end of the study of this course, students should:
i) appreciate the importance of software quality and the essence of software reliability engineering,
ii) be familiar with software design patterns, development process standards, and testing techniques, and
iii) be up-to-date about emergent technologies and practical issues in software engineering.

Prerequisite: COMP3006 Software Engineering, or
COMP3007 Systems Analysis and Design, or
COMP3047 Software Engineering

Course Intended Learning Outcomes (CILOs):
Upon successful completion of this course, students should be able to:

No.	Course Intended Learning Outcomes (CILOs)
	Knowledge
1	Explain software reliability engineering process, techniques, and the applicability.
2	Explain essential recurring structures or patterns in software design and implementation, as well as their applications in developing reliable software systems.
3	Explain software test planning process, testing strategies, and testing techniques.
4	Explain software models and test design patterns.
	Professional Skill
5	Perform software design, development, and testing systematically, following the underlying patterns and essential techniques introduced.

Calendar Description: This course is aimed to further develop students' knowledge and skills in software engineering, and to introduce and discuss software design patterns, state-of-the-art techniques and advanced topics in developing reliable software systems.

Teaching and Learning Activities (TLAs):

CILOs	Type of TLA
1-4	Students will acquire the concepts and develop understandings through lectures and in-class exercises.
5	Students will develop their software design, development, and testing skills through demonstrated software development exercises and projects.

Assessment:

No.	Assessment Methods	Weighting	CILOs to be addressed	Description of Assessment Tasks
1	Continuous Assessment	60%	5	The group project provides opportunities for students to practice and demonstrate their skills and abilities in applying and integrating the principles and techniques of software design, development, and testing learned.
2	Examination	40%	1-4	Final examination questions evaluate students' in-depth knowledge and understanding of the key principles and techniques necessary for developing reliable software systems.

Assessment Rubrics:

Excellent (A)	• Achieves all the five LOs, demonstrating a good mastery of both the theoretical and practical aspects of the knowledge and skills associated with software design, development, and testing • Able to develop and present sound arguments and correct solutions to problems, accompanied by in-depth analysis and insight • Demonstrates a thorough understanding and solid knowledge of the principles and techniques of software design, development, and testing • Able to draw on a variety of techniques and relevant knowledge and appropriately apply them to new software design, development, and testing situations and problems
Good (B)	• Achieves all five LOs, demonstrating a good understanding of the associated concepts and underlying methodologies • Able to develop solutions to problems, accompanied by adequate explanations • Demonstrates a competent level of knowledge of the principles and techniques of software design, development, and testing • Ability to make use of appropriate techniques and knowledge and apply them to familiar situations and problems
Satisfactory (C)	• Achieves most of the five LOs, demonstrating a basic level of understanding of the associated concepts and underlying methodologies • Able to provide acceptable solutions to problems • Demonstrates an adequate level of knowledge of the principles and techniques of software design, development, and testing • Ability to make use of some techniques and knowledge and apply them to familiar situations
Marginal Pass (D)	• Achieves most of the five LOs, with minimal understanding of the associated concepts and underlying methodologies • Able to provide solutions to simple problems • Demonstrates a basic level of knowledge of the principles and techniques of software design, development, and testing • Ability to apply some techniques and knowledge to a limited number of typical situations
Fail (F)	• Achieves less than three of the LOs, with little understanding of the associated concepts and underlying methodologies • Unable to provide solutions to simple problems • Knowledge of the principles and techniques of software design, development, and testing falling below the basic minimum level • Unable to apply techniques or knowledge to situations or problems

Course Content and CILOs Mapping:

Content		**CILO No.**
I	Software Quality	1
II	Software Design and Development	2,4,5
III	Software Design Patterns	2,4
IV	Software Testing	1-4,5

References:

- K. Beck, Implementation Patterns, Addison-Wesley, 2007
- R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Addison-Wesley, 2007
- Glenford J. Myers, Corey Sandler, Tom Badgett, The Art of Software Testing, 3rd Edition, Wiley, Nov., 2011
- Lee Copeland, A Practitioner's Guide to Software Test Design, Artech House, Jan. 2004
- Lisa Crispin & Janet Gregory, Agile Testing: A Practical Guide for Testers and Agile Teams, 1st Edition, Addison-Wesley Professional, Jan. 2009
- Wazlawick, Object-Oriented Analysis and Design for Information Systems, Morgan Kaufman, 2014

- Jeffrey A. Hoffer, Joey George, and Joe A. Valacich, Modern Systems Analysis and Design, (7th Edition), Prentice Hall, 2013
- Alan Dennis, Barbara Haley Wixom, and David Tegarden, Systems Analysis and Design: An Object-Oriented Approach with UML (5th Edition), Wiley, 2015
- Hassan Gomaa, Software Modeling and Design – UML, Use Cases, Patterns, and Software Architectures, Cambridge University Press, 2011
- Jez Humble, and David Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Addison-Wesley, 2010
- Robert V. Binder, Testing Object-Oriented Systems: Models, Patterns, and Tools, Addison-Wesley Professional, 1999
- John Viega, and Gary McGraw, Building Secure Software: How to Avoid Security Problems the Right Way, Addison-Wesley, 2006
- Ivar Jacobson, Grady Booch, and James Rumbaugh, The Unified Software Development Process, Addison-Wesley, 1999
- Joshua Kerievsky, Refactoring to Patterns, Addison-Wesley Professional, 2004
- Cay S. Horstmann, Object-oriented Design and Patterns, John Wiley & Sons, 2006
- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994

Course Content:

<u>Topic</u>

I. Software Quality
   A. Quality control techniques
   B. Software reliability engineering (a development perspective)

II. Software Design and Development
   A. Formal specifications and formal design techniques
   B. Basics of design patterns
   C. Implementation plan
   D. Development tools and code generation
   E. Coding styles and standard practice

III. Software Design Patterns
   A. Creational design patterns
   B. Structural design patterns
   C. Behavioral design patterns

IV. Software Testing
   A. Verification and validation
   B. Design-level and implementation-level testing
   C. Testing plan, strategies and techniques
   D. Testing object-oriented software
   E. Combinational models and state machines
   F. Responsibility-based testing and test design patterns