TQF 3 Course Specifications
Section 1 General Information

1. Course code and course title

Thai ICCS ๓๗๐ การสร้างระบบซอฟต์แวร์

English ICCS 370 Software System Construction

2. Number of credits 4 (4-0-8) (Lecture/Lab/Self-study)
3. Program and type of subject
   - 3.1 Program Bachelor of Science (Computer Science)
   - 3.2 Type of Subject Required course
4. Course Coordinator and Course Lecturer
   - 4.1 Course Coordinator Sunsern CHEAMANUNKUL, PhD
   - 4.2 Course Lecturers TBA
5. Trimester/ Year of Study
   - 5.1 Trimester All trimesters (excluding summer session) / for all students in all International College Undergraduate Programs
   - 5.2 Course Capacity Approximately 30 students
6. Pre-requisite ICCS208 Data Structures and Abstractions
7. Co-requisites -
8. Venue of Study Mahidol University, Salaya Campus

Section 2 Goals and Objectives

1. Course Goals

To familiarize students with the tool chains and the process of modern software development.

2. Objectives of Course Development/Revision

2.1 Course Objectives

This course provides a hands-on experience to modern software development, created using both TQF1 and ACM frameworks as a guideline.

2.2 Course-level Learning Outcomes: CLOs

By the end of the course, students will be able to (CLOs)

1. CLO 1 Recognize the concepts of intellectual property, copyright licenses, and law pertaining to open-source software and libraries.
2. CLO 2 Develop skills in a number of positions involved in the software development, such as the team leader, the designer, and the tester.
3. CLO 3 Deconstruct existing software applications to reveal its structure, components, and process of construction.
4. CLO 4 Develop medium-scale software system from scratch using modern software development tools and technologies: identifying and analyzing the problems to be solved; exploring multi-tier system designs; modelling problems using suitable abstractions, and using refactoring and testing to ensure software quality.

Section 3 Course Management

1. Course Description

Modern development tools, including development environments, build tools, test automation, debugger; Profiling and performance tuning; Software verification; Software testing; UML diagrams; Design patterns; System architectures; Entity design and conceptual modeling; Source code refactoring

เครื่องมือและสิ่งแวดล้อมในการพัฒนาซอฟต์แวร์สมัยใหม่; เครื่องมือในการสร้างชุดคำสั่งจากรหัสต้นฉบับ; การสร้างชุดทดสอบอัตโนมัติ; การตรวจหาและแก้ไขข้อบกพร่องของซอฟต์แวร์; การทำโปรไฟล์และปรับแต่งประสิทธิภาพ; การทวนสอบซอฟต์แวร์; แผนภาพยูเอ็มแอล; แบบแผนและวิธีการออกแบบ; สถาปัตยกรรมระบบ; การออกแบบเอนทิตีและการจำลองแบบเชิงแนวคิด; การรีแฟคทอริ่งรหัสต้นทาง

2. Credit hours per trimester

Lecture (Hour(s))	Laboratory/field trip/internship (Hour(s))	Self-study (Hour(s))
48	0	96

3. Number of hours that the lecturer provides individual counseling and guidance.

1 hour/week

Section 4 Development of Students' Learning Outcome

1. Short summary on the knowledge or skills that the course intends to develop in students (CLOs)

By the end of the course, students will be able to:

1. CLO 1 Recognize the concepts of intellectual property, copyright licenses, and law pertaining to open-source software and libraries.
2. CLO 2 Develop skills in a number of positions involved in the software development, such as the team leader, the designer, and the tester.
3. CLO 3 Deconstruct existing software applications to reveal its structure, components, and process of construction.
4. CLO 4 Develop medium-scale software system from scratch using modern software development tools and technologies: identifying and analyzing the problems to be solved; exploring multi-tier system designs; modelling problems using suitable abstractions, and using refactoring and testing to ensure software quality.

2. Teaching methods for developing the knowledge or skills specified in item 1 and evaluation methods of the course learning outcomes

ICCS 370	Teaching methods	Evaluation Methods
CLO1	Reading assignment, interactive lecture, case studies, quiz, group activities, group discussion	Quiz, Homework, Examination
CLO2	Reading assignment, interactive lecture, case studies, quiz, group activities, group discussion	Quiz, Homework, Examination
CLO3	Reading assignment, interactive lecture, case studies, quiz, group activities, group discussion	Quiz, Homework, Examination
CLO4	Reading assignment, interactive lecture, case studies, quiz, group activities, group discussion	Quiz, Homework, Examination

Section 5 Teaching and Evaluation Plans

1. Teaching plan

Week	Topic	Number of Hours		Teaching Activities/ Media	Lecturer
		Lecture Hours	Lab/Field Trip/Internship Hours		
1-2	Modern development tools, including development environments, build tools, test automation, debugger.	8	0	Reading assignment, interactive lecture, quiz, group activities, case studies, group discussion	TBA
3-4	Design Pattern	8	0		
5-6	Entity design and conceptual modelling using UML.	8	0		
7-8	System architectures and Micro-service design	8	0		
9-11	Software quality control: Source code refactoring, code review, and testing.	12	0		
12	Basic software development methodologies.	4	0		
	Total	48	0		

2. Plan for Assessing Course Learning Outcomes

2.1 Assessing and Evaluating Learning Achievement

a. Formative Assessment

- Worksheet
- Class discussion

b. Summative Assessment

(1) Tools and Percentage Weight in Assessment and Evaluation

Learning Outcomes	Assessment Methods	Assessment Ratio (Percentage)	
CLO 1 Recognize the concepts of intellectual property, copyright licenses, and law pertaining to open-source software and libraries.	Homework & Quiz	10	15
	Examination	5	
CLO 2 Develop skills in a number of positions involved in the software development, such as the team leader, the designer, and the tester.	Homework & Quiz	10	15
	Examination	5	
CLO 3 Deconstruct existing software applications to reveal its structure, components, and process of construction.	Homework & Quiz	20	30
	Examination	10	
CLO 4 Develop medium-scale software system from scratch using modern software development tools and technologies: identifying and analyzing the problems to be solved; exploring multi-tier system designs; modelling problems using suitable abstractions, and using refactoring and testing to ensure software quality.	Homework & Quiz	30	40
	Examination	10	
			100

(2) Grading System

Grade	Achievement	Final Score (% Range)	GPA
A	Excellent	90-100	4.0
B+	Very good	85-89	3.5
B	Good	80-84	3.0
C+	Fairly good	75-79	2.5
C	Fair	70-74	2.0
D+	Poor	65-69	1.5
D	Very Poor	60-64	1.0
F	Fail	Less than 60	0.0

(3) Re-examination (If course lecturer allows to have re-examination)

N/A - (Not applicable with MUIC)

3. Student Appeals

N/A

Section 6 Teaching Materials and Resources

1. Textbooks and/or other documents/materials

- Lecture notes provided by the lecturer
- Online material provided by the lecturer

2. Recommended textbooks and/or other documents/materials

Selected readings from pertinent sources and textbooks or video clips, as posted on the course's e-learning site

3. Other Resources (If any)

N/A

Section 7 Evaluation and Improvement of Course Management

1. Strategies for evaluating course effectiveness by students

   1.1 Student feedback of instructors, teaching methods and materials, and course content through MUIC student evaluation forms

2. Strategies for evaluating teaching methods

   2.1 Evaluation of effectiveness based on student evaluation scores and comments

   2.2 Evaluation through peer observations by co-instructor or other Division faculty

3. Improvement of teaching methods

   3.1 Adjustments based on student feedback, personal observations, comments from peer observations and discussions with supervisor and/or other Division faculty in one-on-one and/or group meetings as specified by MUIC guidelines

4. Verification process for evaluating students' standard achievement outcomes in the course

   4.1 Verification through student performance on assessments based on MUIC/Division standards

5. Review and plan for improving the effectiveness of the course

   5.1 Course instructors (and coordinator/supervisor) will meet to discuss results of student evaluations and student performance based on learning outcomes in order to identify point for improvement

   5.2 Strategy for improvement set according to MUIC/Division guidelines

Appendix
Alignment between Courses and General Education courses

Table 1 The relationship between course and Program Learning Outcomes (PLOs)

	Program Learning Outcomes (PLOs)					
	PLO1	PLO2	PLO3	PLO4	PLO5	PLO6
(ICCS370)		P		I	M	I

Table 2 The relationship between CLOs and Program LOs (Number in table = Sub LOs)

ICCS370	Learning Outcomes in the Computer Science Program					
	1	2	3	4	5	6
CLO 1 Recognize the concepts of intellectual property, copyright licenses, and law pertaining to open-source software and libraries.		2.1				
CLO 2 Develop skills in a number of positions involved in the software development, such as the team leader, the designer, and the tester.				4		
CLO 3 Deconstruct existing software applications to reveal its structure, components, and process of construction.					5.3	
CLO 4 Develop medium-scale software system from scratch using modern software development tools and technologies: identifying and analyzing the problems to be solved; exploring multi-tier system designs; modelling problems using suitable abstractions, and using refactoring and testing to ensure software quality.					5.4	6.1

Table 3 The description of Program LOs and Sub LOs of the course

CS LOs	Sub LOs
PLO2 Carry out work with scientific integrity and professionalism.	2.1 Recognize the concepts of intellectual property, copyright licenses, and law pertaining to information technology.
PLO4 Use a teamwork mindset in the context of computing.	
PLO5 Execute common computing methodologies appropriate for a problem scenario.	5.3 Deconstruct a computer system to reveal its structure, components, and process of construction.
	5.4 Select common computing techniques (e.g., standard algorithms, data structures, design patterns, programing style, and computing paradigms) appropriate for a given problem scenario.
PLO6 Formulate computational solutions to novel situations grounded on the foundation of computer science.	6.1 Model a given problem using suitable abstractions, including problem decomposition, in the context of computing