

CS3343: SOFTWARE ENGINEERING PRACTICE

Effective Term

Semester A 2024/25

Part I Course Overview

Course Title

Software Engineering Practice

Subject Code

CS - Computer Science

Course Number

3343

Academic Unit

Computer Science (CS)

College/School

College of Computing (CC)

Course Duration

One Semester

Credit Units

3

Level

B1, B2, B3, B4 - Bachelor's Degree

Medium of Instruction

English

Medium of Assessment

English

Prerequisites

Nil

Precursors

CS3342 Software Design or equivalent

Equivalent Courses

Nil

Exclusive Courses

Nil

Part II Course Details

Abstract

CS3343 Software Engineering Practice provides students with a practical learning experience to apply software engineering principles and techniques. Working in student groups with real-world role-playing, students will develop a substantial

software system that satisfies specified requirements and quality standards, emulating common software engineering industry practices.

Course Intended Learning Outcomes (CILOs)

CILOs		Weighting (if app.)	DEC-A1	DEC-A2	DEC-A3
1	Apply the principles and techniques of software engineering and modeling, automated software testing and maintenance to ensure high quality of software.		x	x	x
2	Discuss and present software development projects professionally, emphasizing strong project presentation and engagement skills.		x	x	
3	Prepare concise, professional software technical documentation, utilizing industry-standard tools to produce high-quality deliverables.		x	x	
4	Work professionally within a software development team, leveraging tools and techniques to collaborate effectively.		x	x	

A1: Attitude

Develop an attitude of discovery/innovation/creativity, as demonstrated by students possessing a strong sense of curiosity, asking questions actively, challenging assumptions or engaging in inquiry together with teachers.

A2: Ability

Develop the ability/skill needed to discover/innovate/create, as demonstrated by students possessing critical thinking skills to assess ideas, acquiring research skills, synthesizing knowledge across disciplines or applying academic knowledge to real-life problems.

A3: Accomplishments

Demonstrate accomplishment of discovery/innovation/creativity through producing /constructing creative works/new artefacts, effective solutions to real-life problems or new processes.

Learning and Teaching Activities (LTAs)

LTAs		Brief Description	CILO No.	Hours/week (if applicable)
1	Lecture	Students will engage in lectures to gain knowledge about best practices in software engineering, as well as automated and contemporary tools to streamline the development process.	1, 2, 3, 4	
2	Tutorial	Students will participate in tutorial activities to engage with their groups and extend their use of computer software tools for practical software development.	1, 2, 3, 4	

3	Group Project	Students will collaborate in groups to consolidate their learning through the production of a final group report based on a realistic software development project chosen by the team. This exercise will enable students to practice project management skills within a team environment. Each group is expected to define the distinct roles of individual members clearly. The group report will document their project progress, the challenges encountered, the solutions devised, and the personal experiences gained from playing their respective roles in the project.	1, 2, 3, 4	
---	---------------	---	------------	--

Assessment Tasks / Activities (ATs)

ATs	CILO No.	Weighting (%)	Remarks (e.g. Parameter for GenAI use)
1	Practical Test	1, 3	40
2	Group Project	1, 2, 3, 4	30

Continuous Assessment (%)

70

Examination (%)

30

Examination Duration (Hours)

2

Minimum Continuous Assessment Passing Requirement (%)

40

Minimum Examination Passing Requirement (%)

30

Additional Information for ATs

For a student to pass the course, at least 40% of the maximum mark for the continuous assessment and 30% of the maximum mark for the examination must be obtained.

Assessment Rubrics (AR)**Assessment Task**

Practical tests

Criterion

- 1.1 CAPACITY for SELF-DIRECTED LEARNING to understand the tools and practices of software development.
- 1.2 ABILITY to EXPLAIN AND APPLY software testing techniques.

Excellent (A+, A, A-)

High

Good (B+, B, B-)

Significant

Fair (C+, C, C-)

Moderate

Marginal (D)

Basic

Failure (F)

Not even reaching marginal levels

Assessment Task

Group project

Criterion

- 2.1 CAPACITY for SELF-DIRECTED LEARNING to apply the learnt practices to real problems, produce an application and write technical documents.
- 2.2 ABILITY to EXPLAIN AND DEMONSTRATE IN DETAIL about project management in a team environment.
- 2.3 ABILITY to APPLY the software development and testing procedures to produce a quality software system within a team.

Excellent (A+, A, A-)

High

Good (B+, B, B-)

Significant

Fair (C+, C, C-)

Moderate

Marginal (D)

Basic

Failure (F)

Not even reaching marginal levels

Assessment Task

Examination

Criterion

- 3.1 CAPACITY for SELF-DIRECTED LEARNING to understand the tools and practices of software development.

3.2 ABILITY to EXPLAIN AND APPLY software development and testing procedures to produce high quality software systems.

Excellent (A+, A, A-)

High

Good (B+, B, B-)

Significant

Fair (C+, C, C-)

Moderate

Marginal (D)

Basic

Failure (F)

Not even reaching marginal levels

Part III Other Information

Keyword Syllabus

Automated Software Engineering, Software Testing and Maintenance, Software Project Management, Requirement Engineering and Design Solution, Object Oriented Modeling and Design, Coding and Implementation, Technical Documentation, and Project Presentation.

Syllabus

- Automated Software Testing - JUnit, Java Implementation
- Software project management
Project planning and scheduling. Team organisations and role playing. Version control and configuration management. Documentation.
- Requirement elicitation and specification
User requirements specification. Prototyping.
- Design
Input design. Output design. User interface design. Designing for Change. CASE tools.
- System implementation and testing
Integrated software engineering environments (IDE). Programming standards. Software testing strategies. Software Debugging. Refactoring. CASE tools.

Reading List

Compulsory Readings

Title	
1	Nil

Additional Readings

Title	
1	Sommerville I. (2012) Software Engineering. Addison Wesley, 10th edition.
2	Larman C. (2005) Applying UML and Patterns: Introduction to OOA/D and Iterative Development. Pearson Education, Prentice Hall, 3rd edition.
3	Martin R C, and Martin M. (2006) Agile Principles, Patterns, and Practices in C#. Prentice Hall.
4	Hughes B. and Cotterell M. (2005) Software Project Management. McGraw-Hill, 4th edition.

5	Christensen M. and Thayer R.H. (2002) A Manager's Guide to Software Engineering' s Best Practice. Wiley-IEEE Computer Society.
---	--