هيئة تقويم التعليم والتدريب
Education & Training Evaluation Commission
اعتماد
NCAAA
2024
TP-153

# Course Specification

## (Bachelor)

| | |
|---|---|
| **Course Title:** | Program Comprehension |
| **Course Code:** | SE4718 |
| **Program:** | BSc in Software Engineering |
| **Department:** | Software Engineering |
| **College:** | College of Computing |
| **Institution:** | Umm Al Qura University |
| **Version:** | 1.0 |
| **Last Revision Date:** | 22/04/2025 |

# Table of Contents

## A. General information about the course:

### 1. Course Identification

| 1. Credit hours: (3) | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| **2. Course type** | | | | | |
| **A.** | ☐University | ☐College | ☒ Department | ☐Track | ☐Others |
| **B.** | ☐ Required | | ☒ Elective | | |
| **3. Level/year at which this course is offered: ( 3rd year/ 5th or 6th level) or ( 4th year/8th level)** | | | | | |
| **4. Course General Description:** | | | | | |
| A vital skill for a successful software engineer is the ability to work with large, pre-existing code bases. Often, in industry or research projects, you may be tasked with fixing a bug, adding a small feature, or implementing some major changes in pre-existing code bases. As a result, learning how to effectively navigate, understand, and contribute to a large code base can help you be prepared for the demands of a software engineering job. In this course, students will learn about the processes and tools for working with large code bases. We will focus on program comprehension techniques (such as code navigation, diagramming, using a debugger, etc.) and code management workflows (code review, Git, task managers, etc.) used when working with large code bases. | | | | | |
| **5. Pre-requirements for this course** (if any)**:** | | | | | |
| **SE3102 - Software Design and Architecture** | | | | | |
| **6. Co-requisites for this course** (if any)**:** | | | | | |
| N/A | | | | | |
| **7. Course Main Objective(s):** | | | | | |
| By the end of this course, students will:<br>1. Gain hands-on experience in OSS development, including building, debugging, and contributing to real-world projects.<br>2. Navigate and comprehend complex codebases, enabling meaningful participation in collaborative environments.<br>3. Use test cases and debugging tools to ensure software quality and reliability.<br>4. Develop skills in using CI/CD pipelines, profiling, and integrating LLMs into modern development practices.<br>5. Communicate technical solutions effectively through code walkthroughs, fostering teamwork and collaboration. | | | | | |

## 2. Teaching mode (mark all that apply)

| No | Mode of Instruction | Contact Hours | Percentage |
|---|---|---|---|
| 1 | Traditional classroom | 45 | 100% |
| 2 | E-learning | | |
| 3 | Hybrid<br>• Traditional classroom<br>• E-learning | | |
| 4 | Distance learning | | |

## 3. Contact Hours (based on the academic semester)

| No | Activity | Contact Hours |
|---|---|---|
| 1. | **Lectures** | 45 |
| 2. | **Laboratory/Studio** | 0 |
| 3. | **Field** | 0 |
| 4. | **Tutorial** | 0 |
| 5. | **Others (specify)** | 0 |
| **Total** | | 45 |

## B. Course Learning Outcomes (CLOs), Teaching Strategies and Assessment Methods

| Code | Course Learning Outcomes | Code of PLOs aligned with the program | Teaching Strategies | Assessment Methods |
|---|---|---|---|---|
| **1.0** | **Knowledge and understanding** | | | |
| 1.1 | Understand the foundational concepts of Open Source Software (OSS) and development environments. | K1 | Lecture, Exercise | Quiz, Exams, Assignments |
| 1.2 | Learn about code navigation techniques and debugging principles to analyze and fix software issues. | K2 | Lecture, Exercise | Quiz, Exams, Assignments |
| **2.0** | **Skills** | | | |
| 2.1 | Apply debugging techniques and create experimental code | S1 | Lecture, Exercise, Group discussion | Quiz, Exams, Assignments |

| Code | Course Learning Outcomes | Code of PLOs aligned with the program | Teaching Strategies | Assessment Methods |
|---|---|---|---|---|
| | changes for iterative software improvement. | | | |
| 2.2 | Use test cases to validate software functionality and ensure quality assurance. | S3 | Lecture, Exercise, Group discussion | Quiz, Exams, Assignments |
| 2.3 | Contribute to Open Source Software (OSS) by performing code walkthroughs and collaborative development. | S2 | Lecture, Exercise, Group discussion | Quiz, Exams, Assignments |
| 2.4 | Effectively Communicate how a code base works to others. | S4 | Lecture, Exercise, Group discussion | Quiz, Exams, Assignments |
| **3.0** | **Values, autonomy, and responsibility** | | | |
| 3.1 | Recognize the value of collaboration and community contributions in OSS projects. | V2 | Exercise, Group discussion | Quiz, exams, assignments |
| 3.2 | Appreciate the role of CI/CD and testing in maintaining software reliability and efficiency. | V1 | Exercise, Group discussion | Quiz, exams, assignments |
| 3.3 | | | | |

## C. Course Content

| No | List of Topics | Contact Hours |
|---|---|---|
| **1.** | Introduction to OSS, Development Environments, Building | **3** |
| **2.** | Code Navigation | **3** |
| **3.** | Making Experimental Code Changes | **6** |
| **4.** | The Debugger | **6** |
| **5.** | Debugger + Diagramming | **6** |
| **6.** | Using Test Cases | **3** |
| **7.** | Just in Time Learning | **3** |
| **8.** | LLMs 1 | **3** |
| **9.** | LLMs 2 | **3** |
| **10.** | (CI/CD, Profiling, OSS Contributing) | **6** |
| **11.** | Code Walkthrough | **3** |
| **Total** | | **45** |

## D. Students Assessment Activities

| No | Assessment Activities * | Assessment timing (in week no) | Percentage of Total Assessment Score |
|---|---|---|---|
| 1. | Quizzes | | 10 |
| 2. | Projects | 2-14 | 20 |
| 3. | Assignments | 2-14 | 20 |
| 4. | Mid Term | 7 | 15 |
| 5. | Final Exam | 16-17 | 35 |

*Assessment Activities (i.e., Written test, oral test, oral presentation, group project, essay, etc.).

## E. Learning Resources and Facilities

### 1. References and Learning Resources

| | |
|---|---|
| **Essential References** | • Boswell, D., & Foucher, T. (2011). *The art of readable code: Simple and Practical Techniques for Writing Better Code*. "O'Reilly Media, Inc." |
| **Supportive References** | • Winters, T., Manshreck, T., & Wright, H. (2020). Software engineering at Google: Lessons Learned from Programming Over Time. "O'Reilly Media, Inc."<br>• Spinellis, D. (2003). *Code reading: The Open Source Perspective*. Addison-Wesley Professional.<br>• Feathers, M. (2004). *Working Effectively with Legacy Code*. Prentice Hall Professional.<br>• |
| **Electronic Materials** | • University of California, San Diego. (2024). *CSE 190: Large codebases course schedule*. Retrieved from https://cse190largecodebases.github.io/sp24/schedule |
| **Other Learning Materials** | |

### 2. Required Facilities and equipment

| Items | Resources |
|---|---|
| **facilities**<br>(Classrooms, laboratories, exhibition rooms, simulation rooms, etc.) | Traditional Classroom |
| **Technology equipment**<br>(projector, smart board, software) | Multimedia Projector |
| **Other equipment**<br>(depending on the nature of the specialty) | N/A |

## F. Assessment of Course Quality

| Assessment Areas/Issues | Assessor | Assessment Methods |
|---|---|---|
| Effectiveness of teaching | Students | Direct, Indirect |
| Effectiveness of Students assessment | Faculty, Peer reviewer | Direct, Indirect |
| Quality of learning resources | Faculty, Course coordinator | Direct, Indirect |
| The extent to which CLOs have been achieved | Course coordinator, Program management committee | Direct |
| Other | | |

**Assessors** (Students, Faculty, Program Leaders, Peer Reviewers, Others (specify)

**Assessment Methods** (Direct, Indirect)

## G. Specification Approval

| | |
|---|---|
| **COUNCIL /COMMITTEE** | **SOFTWARE ENGINEERING DEPARTMENT COUNCIL** |
| **REFERENCE NO.** | **THE 17TH MEETING FOR THE ACADEMIC YEAR 1446H** |
| **DATE** | **22/04/2025** |