**Curriculum Syllabus for 《Software Quality Assurance and Testing Techniques》**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| English Course Name | Software Quality Assurance and Testing Techniques | | Total Class Hours | | 64 | Credits | 4 |
| Course Code | F126031 | | Theory Teaching Class Hours | | 32 | Applicable Majors | Software Engineering |
| Course Type (Please marked the course is elective or compulsory) | General Knowledge Courses |  | Practical Teaching Class Hours | Experiments Class Hours | 0 | Prerequisite Courses | Java Programming  Software Engineering  Database Principles |
| Basic Courses |  | Computer Practice Class Hours | 32 | Course College | Computer Science |
| Specialty Courses | Compulsory | Others | 0 | Grass-roots Teaching Organization | Software Engineering Teaching Group |

**I. Course Introduction**

This course is compulsory for undergraduate students that major in software engineering, and it is the origin for them to systematically learn the basic knowledge and professional skills that are related to software quality assurance and testing. Since the ability of software testing has been recognized as one of the preliminary skill that a software engineer should have, more and more people pay great attention to it. Hence, this course has also become one of the core courses for students in software engineering major.

**II.Teaching Objectives**

The objective of this course is to help students to master the basic concepts of software quality assurance and testing, and to learn more about the principles of software quality assurance and testing. Additionally, the ability of analyzing and solving the problem that the students may encounter will also be trained through learning this course.

（1）Master the details of different testing methods, and designing test cases for different scenarios by combining the walkthrough and code inspection. Know the basic concepts and standards of software quality assurance. Be familiar with the document composition in different phases during the software quality assurance.

（2）Learn and master equivalence partitioning, boundary value analysis, causality diagram, and other black-box testing methods, and know their focuses, limitations and suitable scenarios. Applying them to practical projects freely.

（3）Understand the coverage of statements, judgements, conditions and execution paths, and can freely apply the control flow analysis and cyclomatic complexity to design the test cases from the viewpoint of white-box methods.

（4）Learn the main idea of non-functional testing, e.g., unit testing where mock techniques are used, performance test, and integration test. Know how to use Junit, Jprofiler, Jmeter and other tools to perform above tests and data collection. Finally, the comprehensive analysis on complex projects can be conducted by using the mixture of black-box, white-box methods and other testing techniques.

**III. Course Contents and Class Hours Distribution**

**1.Arrangement for Theory Teaching**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number | Chapters or knowledge points (modules) | Course Contents | Class Hours Distribution | Teaching Requirements (The key points, difficulties and methods of teaching should be clarified) | Students’ Task | |
| Assignment Requirements | Other Requirements (Self-study / Discussion) |
| 1 | Basic concepts and methods of software testing | 1. The concept of software quality 2. Software bugs 3. Basic methods of software testing 4. The classification and phases of software testing 5. The scope of software testing. 6. Static testing: code inspection, walk-through, review 7. The process of software quality assurance. | 8 | Know the basic concepts and relations of software quality assurance and software testing, know the basic testing methods and phases. | Select the project for testing based on group | Discuss and determine the test plan. |
| 2 | Black-box test use case design | 1. Equivalence partioning 2. Boundary value 3. Decision table 4. Causality digram | 6 | Master black-box test use case design methods and apply them in real projects. | Designing black-box test cases for the selected project. |  |
| 3 | White-box test use case design | 1. Statement coverage 2. Judgement coverage 3. Condition coverage 4. Condition combination coverage 5. Independent path coverage | 6 | Understand the coverage of statements, judgements, conditions and execution paths, and can freely apply the control flow analysis and cyclomatic complexity to design the test cases from the viewpoint of white-box methods | Designing white-box test cases for the selected project. |  |
| 4 | Unit and integration testing | 1. The definition, objective and task of unit testing. 2. Debug and evaluation 3. Mock technique for unit testing 4. Integration strategies | 4 | Master unit testing and is able to use mock or stub techniques to perform testing. Know the pros and cons of the integration strategies. | Perform unit and integration tests for the selected project. | Learn mock techniques on spring and other advanced frameworks. |
| 5 | Non-functional testing | 1. Pressure testing 2. Performance testing 3. Safety, reliability testing 4. Acceptance testing 5. Software localization | 4 | Know the basic concepts and content of the non-functional testing. Master the techniques and tools for performance testing, and apply them to practical projects. | Perform non-functional testing on selected project. | Independently learn security, acceptance, and localization testing. |
| 6 | Software quality assurance process | 1. Automatic software testing 2. Building testing team and environment 3. Bug tracking and analyzing 4. Project management for software testing 5. Testing report composition | 4 | Understand the basic concepts and tools of automatic software testing. Understand the whole process management of software quality assurance. Master the writing specification of quality analysis report. |  | Study JIRA independently |

**2.Practical Teaching Arrangement**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Number | Project Name | Class Hours | Type | Numbers of People in Each Group | Teaching Requirements (The key points, difficulties and methods of teaching should be clarified) | Students’ Task | |
| Assignment Requirements | Other Requirements (Self-study / Discussion) |
| 1 | Unit testing tool usage | 4 | Design | 1 | Performing Junit testing based on the designed test cases | Finish Junit based unit testing |  |
| 2 | Senior unit testing | 4 | Design | 1 | Using EasyMock tools to perform mock based unit testing. | Using EasyMock to test and complete the report |  |
| 3 | Integration testing | 4 | Design | 1 | Performing integration testing based on unit testing and mock technqiues | Complete the struts and other framework unit testing and finish the report | Unit testing on other frameworks |
| 4 | Performance testing | 4 | Design | 3-7 | Performance testing based on Jprofiler | Complete JProfiler testing and the report |  |
| 5 | Pressure testing | 4 | Design | 3-7 | Pressure testing based on JMeter | Complete JMeter testing and the report |  |
| 6 | Automatic testing | 4 | Design | 1 | Completing the automatic testing based HP QTP tool | Complete QTP testing and the report |  |
| 7 | Testing process management | 4 | Integrated | 3-7 | Using JIRA to manage the whole process of software testing. |  | Learn JIRA independently |
| 8 | Testing report composition | 4 | Integrated | 3-7 | Composing the final testing report based on the specification | Finish the final report |  |

**IV. Assessment Methods and Performance Evaluation Methods**

The assessment of this course focuses on the procedure. The final score consists of two main parts, i.e., the formative score and the score for the final exam, and each of them accounts for 50% of the final score. Note that, formative score mainly refers to the evaluation on the homework, experiments and discussions. Specifically, the evaluation ratio and basic requirements for formative score are detailed as follows:

（1）homework + the project report：accounts for 80% of the formative score. Requirements：there will be at least one homework assignment and a project report, and to finish the analysis report the students are required to learn some extra knowledge by surveying technical reports. Moreover, each report for the experiments must be included into the final report. Through composing this report, the ability engineering problem analysis, experiment evaluation design and the report composition will be improved.

（2）Discussion: accounts for 20% of the formative score. Requirement: the teacher organizes at least one discussion, and mainly examine the autonomous learning ability, teamwork ability, language expression and communication skills of the student.

（3）The final exam, which accounts for 50% of the final score, is a closed-book one.

**V. Textbooks, Course Websites and Bibliographies**

Textbooks： *Foundations of Software Testing, Aditya P.Mathur, Addison-Wesley Professional, February 13, 2014, 2 edition*

Course Websites in Online Teaching Platform (Core courses must be filled):

Bibliographies:*【1】The art of software testing, Glenford J.Myers, Wiley, November 8, 2011,3 edition*

*【2】Software Testing, Ron Patton, Sams Publishing, August 5, 2005, 2 edition*

**Writer: Bin Cao**

**Reviewer: Sheng Liu, Xianzhong Tian**

**Teaching Team Members of The Course: Bin Cao, Ting Wang**