CONESTOGA

COURSE OUTLINE

Course Programming: Software Quality Assurance (2014-2015)

Code / Version PROG2070 (101)

Total Hours 45

Credits 3

PreRequisite(s) PROG2370 (100) Object Oriented Game Prog

CoRequisite(s)

Course Description

Software quality is everybody's job, no matter what their role in a software product development or IT organization. In this course, we will explore various approaches to testing and improving software quality. We will also learn techniques, practices and tools that programmers employ to achieve higher quality of software products.

PLAR Eligible: Yes

Course Outcomes

Successful completion of this course will enable the student to:

1. Apply software quality methodologies and best practices to all phases of application development, from requirements, to implementation, and to testing.
2. Design a performance test plan that is representative of an application's behaviour and avoids measurement bias.
3. Develop unit tests and practice test-driven development (TDD) using case studies of abstract data type (ADT) implementations.
4. Develop system integration tests against web-based and/or database applications using various tools.
5. Assess the overall quality of a test suite using test coverage analysis and mutation testing techniques.
6. Explain the use of software development infrastructure elements, such as version control, build scripting, and continuous integration, and how these elements promote quality throughout the entire software development process.
7. Explain the goals and classification of various testing strategies and the limits of applicability of each strategy.

Unit Outcomes

Successful completion of the following units will enable the student to:

1.0 Software quality standards and models

- 1.1 Define software quality and software quality assurance
- 1.2 Differentiate between unit, integration, system and acceptance testing
- 1.3 Explain the concepts of defects, faults, errors, and failures
- 1.4 Describe the significant characteristics of software quality models such as ISO 9000-9001

2.0 Software development infrastructure

- 2.1 Describe how a version control system is used in software development
- 2.2 Describe the tradeoffs of using continuous integration during software development
- 2.3 Identify the metadata required to monitor testing infrastructure effectively
- 2.4 Develop software failure reports using a standard reporting template

3.0 Unit testing

- 3.1 Use a unit-testing framework such as NUnit to develop and run unit tests
- 3.2 Organize test and production code effectively in a software solution to permit effective testing
- 3.3 Apply guidelines for valuable, effective unit tests

3.4 Differentiate good unit tests from bad ones and non-unit tests

3.5 Develop software in a test-first (test-driven) fashion

4.0 Economics of software quality

4.1 Describe various economic factors as they relate to software quality, including software quality metrics

4.2 Describe the efficiency and effectiveness of various testing strategies

4.3 Analyze a software component using measures such as cyclomatic complexity to determine the most effective testing strategies

5.0 Managing testing infrastructure

5.1 Analyze code test coverage using code coverage analysis tools

5.2 Improve test coverage with unit tests

5.3 Utilize mutation testing techniques to assess the effectiveness of a test suite

6.0 Testing strategies

6.1 Appreciate communication breakdowns in requirements and design that can lead to poor software quality

6.2 Differentiate system, integration, and acceptance testing

6.3 Estimate the resources required for testing using industry-accepted averages

7.0 Functional testing

7.1 Describe the principles and the role of functional testing in a development project

7.2 Apply functional testing techniques using boundary condition analysis, decision tables, and/or randomized testing as applicable.

8.0 Performance testing

8.1 Explain common pitfalls in conducting performance and scalability testing

8.2 Describe the most important factors that impact the performance of a typical database application

8.3 Describe factorial and partial factorial test designs to effectively test a combination of performance factors

9.0 Integration testing

9.1 Develop integration tests interacting with Web-based user interfaces (UI)

9.2 Describe techniques to perform integration tests of an application's data access layer (DAL)

10.0 Testing database applications

10.1 Discuss the common characteristic errors of SQL applications

10.2 Explain the issues of testing database applications with respect to performance and scale, and the problem of measurement bias

10.3 Apply mutation testing techniques to SQL programs

10.4 Explain the fundamental issues of test data generation for database applications

Required Student Resources

Gerald D. Everett and Raymond McLeod Jr.. Software Testing (2007). John Wiley and Sons (IEEE Press).

Optional Student Resources

Kshirasagar Naik and Priyadarshi Tripathy. Software Testing and Quality Assurance: Theory and Practice (2008). John Wiley and Sons.

Capers Jones and Olivier Bonsignour. The Economics of Software Quality (2012). Addison-Wesley.

Yogesh Singh. Software Testing (2012). Cambridge University Press.

Roy Osherove. The Art of Unit-Testing: With Examples in .NET (2009). Manning Publications.

Course	Programming: Software Quality Assurance (2014-2015)
Code / Version	PROG2070 (101)

Evaluation

The minimum passing grade for this course is 55 (D).

In order to successfully complete this course, the student is required to meet the following evaluation criteria:

Assignments	35.00
Final Examination	40.00
Midterm Examination	25.00
	100.00 %

Other

Conestoga College is committed to providing academic accommodations for students with documented disabilities. Please contact the Accessibility Services Office.

The policies and procedures of Conestoga College and the School of Engineering and Information Technology apply to this course.

Prepared By Glenn Paulley

School Information Technology

Date 2014-10-06