Module: Software Testing 281

Module name:	Software Testing 281
Code:	SWT281
NQF level:	6
Type:	Elective – Bachelor of Computing (all streams)
Contact time:	34
Structured time:	6
Self-directed time:	40
Notional hours:	80
Credits:	8
Prerequisites:	PRG282

Purpose

The main focus of this course is on realistic, pragmatic steps for rigorous and organized software testing. It clarifies testing terminology and covers the different types of testing performed at each phase of the software lifecycle together with the issues involved in these types of testing. The course will discuss how tests can be derived from requirements and specifications, design artefacts, or the source code, and introduce proper testing tools. At the end of the course, students will have an appreciation of a range of testing techniques, and an understanding of rigorous testing theory. They will be able to select an appropriate testing strategy, devise suitable test cases, and formulate correctness hypotheses.

Outcomes

Upon successful completion of this module, the student will be able to demonstrate:

- Detailed knowledge of the main areas of software testing, including an understanding of and the ability to apply the key terms, concepts, standards, principles and theories of software testing to new but relevant contexts
- Knowledge of specific testing levels and testing techniques and how that knowledge relates to other stages of the software development lifecycle, including the use of suitable software development and testing tools, with a sound understanding of each tool's extent of applicability and capabilities
- The ability to evaluate, select and apply appropriate testing models, processes and practices, and strategies in the context of the software development lifecycle, for both complete lifecycles and individual phases of the lifecycle
- The ability to plan and apply the appropriate level of testing within the context of a software development application, gathering evidence and applying solutions, based on evidence and appropriate software testing procedures, to meet the requirements of the project beneficiaries
- The ability to evaluate different sources of information, to select information appropriate to the task and to design specific and measurable test cases to ensure coverage and traceability to requirements
- The ability to present and communicate complex information regarding testing results and project status reliably and coherently using appropriate professional problem reporting

techniques, metrics, and testing status reports to colleagues, managers, and end users by producing appropriate documentation for managing each stage of the testing process
- The ability to make decisions and act appropriately in familiar and in new testing contexts, demonstrating an understanding in the relation between software systems, and of how actions or developments in one system impact on other systems
- The ability to work effectively in a team and to take responsibility for their decisions and actions and the decisions and actions of others within the context of this team, including the responsibility for the use of resources where appropriate

Assessment

- Continuous evaluation of theoretical work through written assignment, a formative, and a summative test.
- Continuous evaluation of project work, where the student must design, manage and report on the evaluation of testing methodologies and the selection of an appropriate methodology for a given scenario, justifying the choice made with well-formed arguments and evidence.
- Final assessment through a written examination.
- The assignments or projects collectively will count 30% of your class mark.
- All tests will collectively account for 70% of your class mark.
- Your class mark contributes 30% towards your final mark for the subject, while the final assessment accounts for 70% of your final mark.

Teaching and Learning

Learning materials

Prescribed books (EBSCO)

- 🕮 Rex Black et al. (2017) Agile Testing Foundations : An ISTQB Foundation Level Agile Tester Guide. Swindon, UK: BCS, The Chartered Institute for IT.
- 🕮 Hass, A. M. J. (2014) Guide to Advanced Software Testing. Boston: Artech House.
- 🕮 Rex Black (2011) Pragmatic Software Testing : Becoming an Effective and Efficient Test Professional. Chichester: Wiley.
- 🕮 Kshirasagar Naik and Priyadarshi Tripathy (2008) Software Testing and Quality Assurance : Theory and Practice. Hoboken, N.J.: Wiley-Spektrum.
- 🕮 Satheesh Kumar, N. and Subashni S. (2013) Software Testing Using Visual Studio 2012. Birmingham, UK: Packt Publishing. Learning activities

Learning activities

The teaching and learning activities consist of a combination of formal lectures on theoretical concepts, exercises and discussions. These discussions are dialogue-oriented to stimulate peer discussion on practice. One mandatory assignment and one project must be completed during the course. The experiences and progress on these practical components form the content of class discussions.

Notional learning hours

List the learning activities in hours

Activity	Units	Contact Time	Structured Time	Self-Directed Time
Lecture		27.0		13.0

Formative feedback		3.5		
Project	1	3.5		9.0
Assignment	1			3.0
Test	2		4.0	8.0
Exam	1		2.0	7.0
		34.0	**6.0**	**40.0**

Syllabus

- Fundamentals of testing and the test process
- Testing in the Software Life Cycle: the general V-Model, and test levels, including: unit testing; integration testing; system testing; acceptance testing; performance, stress, and configuration testing;
- Static Testing and static analysis tools
- Dynamic analysis and test design techniques, including Black Box testing, White Box testing
- Test management, including: test planning, organisation and strategy, test progress monitoring and control, test reporting
- Testing Tools: Types and selection of tools