Knight Foundation School of Computing and Information Sciences

Course Title: Fundamentals of Software Testing **Date:** 05/27/2020

Course Number: CEN 4072

Number of Credits: 3

Subject Area: Software Engineering	**Subject Area Coordinator:** Monique Ross **email:** moross@cs.fiu.edu
Catalog Description: Fundamentals of software testing. Topics include: test plan creation, test case generation, program inspections, specification-based and implementation-based testing, GUI testing, and testing tools.	
Textbook: Aditya P. Mathur. "Foundations of Software Testing" 2014 Edition 2, Pearson, ISBN 9788131794760.	
References: The Art of Software Testing, Second edition. Glenford J. Myers. Revised and updated by Tom Badgett and Todd M. Thomas, with Corey Sandler. John Wiley and Sons, New Jersey, U.S.A., 2004. ISBN: 0-471-46912-2	
Prerequisites Courses: COP 3530	
Corequisites Courses: Includes a closed lab component	

Type: Elective for Computer Science Track (Applications group);
Required for Software Design and Development Track

Prerequisites Topics:

- Significant Java or C++ programming experience

Course Outcomes:

1. Be familiar with creating, evaluating and implementing a test plan for a medium-size code segment.
2. Be familiar with program inspections.
3. Be familiar with different types and levels of software testing.
4. Master techniques used to perform specification-based testing and implementation-based testing on programs.
5. Be familiar with the techniques that apply test adequacy coverage criteria to the implementation model.
6. Be familiar with tools to support unit and GUI testing, and code coverage criteria.
7. Be familiar with the limitations of testing

Knight Foundation School of Computing and Information Sciences
CEN 4072
Fundamentals of Software Testing

Relationship between Course Outcomes and Program Outcomes

BS in CS: Program Outcomes	Course Outcomes
a) Demonstrate proficiency in the foundation areas of Computer Science including mathematics, discrete structures, logic and the theory of algorithms	4, 5
b) Demonstrate proficiency in various areas of Computer Science including data structures and algorithms, concepts of programming languages and computer systems.	4, 5
c) Demonstrate proficiency in problem solving and application of software engineering techniques	
d) Demonstrate mastery of at least one modern programming language and proficiency in at least one other.	
e) Demonstrate understanding of the social and ethical concerns of the practicing computer scientist.	
f) Demonstrate the ability to work cooperatively in teams.	1, 2
g) Demonstrate effective communication skills.	1

Assessment Plan for the Course & how Data in the Course are used to assess Program Outcomes

Student and Instructor Course Outcome Surveys are administered at the conclusion of each offering, and are evaluated as described in the School's Assessment Plan: http://www.cis.fiu.edu/programs/undergrad/cs/assessment/

Knight Foundation School of Computing and Information Sciences
CEN 4072
Fundamentals of Software Testing

Outline

Topic	Number of Lecture Hours	Outcome
1. Testing process 1.1. Validation and verification 1.1. Types of testing 1.2. Levels of testing 1.3. Limits of testing 1.4. Importance of testing tools	7	1,3,4,5,6,7
2. Test Plan 2.1. Creation 2.2. Evaluation 2.3. Implementation	7	1,3,4,5
3. Specification-based (black-box) testing 3.1. Equivalence class testing 3.2. Boundary value testing 3.3. State-transition testing 3.4. Domain analysis testing	10	4,5,6
4. Implementation-based (white-box) testing 4.1. Program inspections 4.2. Control flow testing 4.3. Data flow testing 4.4. Mutation testing	10	2,3,4,5,6
5. Testing tools 5.1. Unit testing 5.2. Stubbing/Mocking 5.3. GUI testing	6	5,6

Knight Foundation School of Computing and Information Sciences
CEN 4072
Fundamentals of Software Testing

Learning Outcomes: (Familiarity → Usage → Assessment)

Testing Process

1. Distinguish between program validation and verification. [Familiarity]
2. Describe and distinguish among the different types and levels of testing (unit, integration, systems, and acceptance). [Familiarity]
3. Describe the process to select good regression tests. [Familiarity]
4. Discuss the limitations of testing in a particular domain. [Familiarity]
5. Discuss the issues involving the testing of object-oriented software. [Usage]
6. Describe the process of estimating the number of faults in a small software application based on fault density and fault seeding. [Familiarity]

Creating, Evaluating and Implementing a Test Plan

1. Describe the various components of a test plan [Familiarity]
2. Undertake, as part of a team activity, the planning and testing of a medium-size code segment. [Usage]
3. Undertake, as part of a team activity, creating and documenting a set of tests for a medium-size code segment. [Usage]
4. Undertake, as part of a team activity, presenting and communication a test plan to a project team. [Assessment]

Specification-based Testing

1. Describe the process of developing test cases using multiple techniques based on the specification of the software application. [Familiarity]
2. Create unit, subsystem and system test cases based on the specification of a medium-sized software project [Usage]
3. Evaluate a test suite for a medium-size code segment based on the specification. [Usage]

Implementation-based Testing

1. Describe the process of inspecting a program module. [Familiarity]
2. Describe the process of developing test cases using multiple techniques based on the implementation of the software application. [Familiarity]
3. Enhance unit, subsystem and system test cases based on the implementation of a medium-sized software project [Usage]
4. Evaluate a test suite for a medium-size code segment based on implementation-based adequacy test criteria. [Usage]

Testing Tools

1. Describe the role that tools can play in the validation of software. [Familiarity]
2. Undertake, as part of a team activity, the use of testing tools (unit, system and mocking (stubbing)) to automate the testing of a medium-size software application. [Usage]
3. Demonstrate the automated execution of unit, subsystem and system test suites for a medium-size software application. [Assessment]

Knight Foundation School of Computing and Information Sciences
CEN 4072
Fundamentals of Software Testing

Course Outcomes Emphasized in Laboratory Projects / Assignments

	Outcome	Number of Weeks
1	Software Test Documents Outcomes: 1,3,4,5	4
2	Performing specification-based testing Outcomes: 1,4,6	2
3	Performing implementation-based testing Outcomes: 1,2,5,6	2
4	Performing unit and GUI testing Outcomes: 5,6	2

Oral and Written Communication

Number of written reports: 2 (Software Test Documents)

Approximate number of pages for each report: 80

Number of required oral presentations: 2 (Summary of test document and demonstration of tools to support testing)

Approximate time for each presentation: 25 minutes per group (5 minutes per student)

Social and Ethical Implications of Computing Topics

No significant coverage

Approximate number of class hours devoted to fundamental CS topics

Topic	Core Hours	Advanced Hours
Algorithms:		**1.0**
Software Design:		**0.5**
Computer Organization and Architecture:		
Data Structures:		**1.0**
Concepts of Programming Languages		**0.5**
Other CS Topics:		

Knight Foundation School of Computing and Information Sciences
CEN 4072
Fundamentals of Software Testing

Theoretical Contents

Topic	Class time
Relations	0.5

Problem Analysis Experiences

1.	Control flow analysis of programs
2.	Data flow analysis of programs

Solution Design Experiences

1.	Design of test cases

The Coverage of Knowledge Areas and Units within Computer Science Body of Knowledge[1]

KA	KU	Topic	Type	Lecture Hours
SE	Software Project Management	Team participation • Team processes including responsibilities for tasks, meeting structure, and work schedule • Roles and responsibilities in a software team • Team conflict resolution • Risks associated with virtual teams (communication, perception, structure) Outcome: #1, #2, #3,	Core Tier 2	6
SE	SE3: Software Verification and Validation	[Core-Tier2] • Verification and validation concepts • Inspections, reviews, audits • Testing types, including human computer interface, usability, reliability, security, conformance to specification (cross-reference IAS/Secure Software Engineering) • Testing fundamentals (cross-reference SDF/Development Methods)	Core Tier 2	34

[1]See https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf for a description of Computer Science Knowledge units

		○ Unit, integration, validation, and system testing ○ Test plan creation and test case generation ○ Black-box and white-box testing techniques ○ Regression testing and test automation • Limitations of testing in particular domains, such as parallel or safety-critical systems [Elective] • Object-oriented testing; systems testing Outcomes: #1, #2, #3, #4, #5, #6, #7, #9, #10, #13		