GUJARAT TECHNOLOGICAL UNIVERSITY

Program Name: Master of Computer Applications
Level: Post Graduate
Course / Subject Code: MC03094141
Course / Subject Name: Software Testing

w. e. f. Academic Year:	2024-25
Semester:	3
Category of the Course:	Elective Group - 2

Prerequisite:	• Basic understanding of software development life cycle • Introductory knowledge of software testing • Familiarity with object-oriented programming concepts
Rationale:	Software testing is a crucial phase of the software development lifecycle. This course emphasizes advanced test design techniques with a strong foundation in graph-based, logic-based, and input domain modeling techniques. Learners are exposed to the theoretical and practical applications of automated testing strategies in various domains such as object-oriented and web-based applications, enabling them to deliver reliable and fault-tolerant systems. **Course Pedagogy:** The course will follow an open-source-first teaching model emphasizing hands-on learning. Students will use only open-source tools for test design, implementation, and evaluation. Case studies and coding assignments will be based on real-world problems and public repositories. Tool-based exploration and collaborative lab work will enhance understanding of test strategies.

Course Outcome:

After Completion of the Course, students will be able to:

No	Course Outcomes	RBT Level
01	Apply graph-based testing methods to ensure structural and data flow coverage.	AP
02	Analyze logic-based test criteria and apply them to specification models.	AN
03	Design effective test cases for object-oriented and web-based applications.	CR
04	Evaluate symbolic and concolic testing techniques for program analysis.	EL
05	Construct test cases using mutation, syntax-based and input partitioning methods.	CR

**Revised Bloom's Taxonomy (RBT)*

Teaching and Examination Scheme:

Teaching Scheme (in Hours)			Total Credits L+T+ (PR/2)	Assessment Pattern and Marks				Total Marks
				Theory		Tutorial / Practical		
L	T	PR	C	ESE (E)	PA / CA (M)	PA/CA (I)	ESE (V)	
2	0	2	3	70	30	20	30	150

Course Content:

Unit No.	Content	No. of Hours	% of Weightage
1.	Graph-based testing: Structural coverage (node, edge, path), Data flow coverage using open-source tools like GraphWalker.	9	20
2.	Graph coverage in source code, design and requirements. Logic-based testing using JaCoCo, CodeCover.	9	20
3.	FSM-based testing with GraphWalker, specification-based testing, input space partitioning with ACTS and PICT.	9	20
4.	Syntax-based testing using ANTLR and Grammarinator, mutation testing using PIT and MutPy.	9	20
5.	Symbolic testing using KLEE, concolic testing using CREST and Angr, application to OO and web applications using JUnit and Selenium.	9	20
	Total	**45**	**100**

Suggested Specification Table with Marks (Theory):

Distribution of Theory Marks (in %)					
R Level	**U Level**	**A Level**	**N Level**	**E Level**	**C Level**
10	20	25	15	15	15

Where R: Remember; U: Understanding; A: Application, N: Analyze and E: Evaluate C: Create (as per Revised Bloom's Taxonomy)

References/Suggested Learning Resources:

(a) Reference Books:

1. Paul Ammann & Jeff Offutt, "Introduction to Software Testing", Cambridge University Press.
2. Aditya P. Mathur, "Foundations of Software Testing", Pearson Education.
3. Mauro Pezze & Michal Young, "Software Testing and Analysis: Process, Principles, and Techniques", Wiley.
4. Andreas Zeller, "Why Programs Fail: A Guide to Systematic Debugging", Elsevier.
5. Robert Binder, "Testing Object-Oriented Systems: Models, Patterns, and Tools", Addison-Wesley.

(b) Online Learning Resources and Opensource software and website:

- NPTEL Software Testing: https://nptel.ac.in/courses/106105150/
- GraphWalker: https://graphwalker.github.io/
- PIT (Java): https://pitest.org/
- MutPy (Python): https://mutpy.readthedocs.io/

- JaCoCo: https://www.eclemma.org/jacoco/
- KLEE: https://klee.github.io/
- ANTLR: https://www.antlr.org/
- Selenium: https://www.selenium.dev/
- Playwright: https://playwright.dev/
- ACTS (NIST): https://csrc.nist.gov/projects/automated-combinatorial-testing-for-software

Suggested List of Practical

1. Design graph-based test cases using GraphWalker.
2. Apply data flow test techniques and measure coverage using JaCoCo.
3. Derive logic-based test cases and validate with CodeCover.
4. Construct FSM models and generate tests using GraphWalker.
5. Model input domains using ACTS/PICT and generate test combinations.
6. Perform mutation testing using PIT (Java) and MutPy (Python).
7. Develop symbolic test cases using KLEE.
8. Implement concolic testing using CREST or Angr.
9. Write unit tests for OO programs using JUnit or PyTest.
10. Perform web application testing using Selenium or Playwright.

List of Active Learning Assignments:

- Evaluate and compare test coverage tools (e.g., JaCoCo vs. Coverage.py)
- Analyze test effectiveness in open-source projects.
- Mutation testing on student-developed code using PIT or MutPy.
- Syntax-based test case generation using ANTLR and Grammarinator.

CO- PO Mapping:

Semester 3	**Course Name : Containerization**							
	POs							
Course Outcomes	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8
CO1	3	2	2	2	-	-	-	-
CO2	2	3	2	3	-	-	-	-
CO3	2	2	3	3	-	-	-	-
CO4	2	2	2	3	-	-	-	-
CO5	2	2	3	2	-	-	-	-

*Legend: '**3**' for high, '**2**' for medium, '**1**' for low and '**-**' for no correlation of each CO with PO.*

Note: The CO-PO mapping is indicative; the institute/faculty member can change as required.

* * * * * * *