Software Construction

Code	Credit Hours
SE- 314	2-1

Course Description

This course introduces fundamental concepts and techniques in software development, that is, how to write code that is safe from bugs, easy to understand and ready for change. It will cover fundamental concepts such as specifications and invariants; testing; abstract data types; design patterns for object-oriented programming; concurrent programming and concurrency; and functional programming. The objective of this course is to familiarize students with some of the widely utilized advanced concepts and frameworks in software development. Particularly, it will help students in modelling, design, implementation, and reasoning about solutions using test-first and debug-first programming.

Text Book:

1. Clean Code: A Handbook of Agile Software Craftsmanship, Robert C. Martin, Prentice Hall, latest edition

Reference Book:

1. Engineering Software as a Service: An Agile Approach using Cloud Computing, Latest Edition, A. Fox, D. Petterson
2. Client-Centered Software Development, Allen B. Tucker, CRC Press, latest edition.
3. Software Essentials: Design and Construction, Adiar Dingle, CRC Press, latest edition

Prerequisites

MECH 204 (Material Science and Engineering)

ASSESSMENT SYSTEM FOR THEORY

Quizzes	10%
Assignments	10%
Mid Terms	30%
ESE	50%

ASSESSMENT SYSTEM FOR LAB

Lab Assignments	70%
Semester Project	30%

Teaching Plan

Week No	Topics	Learning Outcomes
1	Introduction	Course Outline, objectives, teaching plan, assessment method, concepts review
2	Static Checking and Testing	Static Checking Validation
3	Testing	Test-first Programming Blackbox and Whitebox testing Choosing Test Cases by Partition Unit Testing
4	Specifications	Specifications Designing Specifications Preconditions or Postconditions Declarative vs. Operational Specs
5	Avoid Debugging	Avoiding Debugging Assertions Localizing Bugs
6	Mutability	Mutability. Risks of Mutation Mutations and Contracts
7	Recursion	Recursion Choosing the Right Recursive Subproblem Structure of Recursive Implementations Common mistakes in Recursive Implementations
8	Debugging	Reproduce the Bug Understand the location and cause of the Bug Fix the Bug
9	**MID TERM IN WEEK 9**	

10	Abstract Data Types	What Abstraction Means Designing Abstract Types Realising ADT Concepts in Java
11	Abstraction Functions	Abstraction Functions and Rep Invariants ADT Invariants replace Preconditions Testing and Abstract Data Type
12	Recursive Datatypes	Recursive Data Types Regular Expressions Grammars
13-14	Parsers generators and Parse tree	Parser Generators An Antlr Grammar Generating the Parser Traversing the Parse Tree Constructing an abstract Syntax Tree Handling Errors
15	Concurrency	Concurrency Two models for Concurrent Programming Race Condition Concurrency is hard to Test and Debug
16	Little languages and Team Version control	Little languages Representing Code as Data Building languages to solve problems Team Version Control Viewing Commit History Using version control as a Team
17	Project Presentation	
18	**End Semester Exam**	

Practical:

Experiment No	Description
1	Lab 01: UML Diagrams
2	Lab 02: Introduction to EclEmma coverage tool and JUnit testing tool
3	Lab 03: Unit Testing and Git
4	Lab 04: Test-First Programming I (Specifications and Unit Tests)
5	Lab 05: Code Review
6	Lab 06: Test First Programming II
7	Lab 07: Abstract Data Type I
8	Lab 08: Abstract Data Type II
9	Lab 09: Representing Expressions
10	Lab 10: Parser Generator Tool ANTLR
11	Lab 11: Parser Generator Tool ANTLR -II
12	Lab 12: Parsing Real Grammers
13	Lab 13: Open Ended Lab
14	Lab 14: Bison Parser
15	Lab 15: Project Presentation/Demos/Vivas
16	Lab 16: Project Presentation/Demos/Vivas