**Software Engineering Excellence (NTAGL-305)**

Duration: 10 days  
Available in Python, C#, Java, C++, or JavaScript

DescriptionA diagram of a code

Description automatically generated

This intensive hands-on course will teach you how to integrate Agile Intensive hands-on course focusing on integrating Agile Development, Test Driven Development (TDD), Object Oriented Principles and Practices, Design Patterns, and Lightweight Design. There’s a strong emphasis on leveraging best practices to improve software craftsmanship and deliver highly valuable software.

The course includes:

* Teams of 2-3 developers engage in a Socratic learning approach, practicing Agile principles by collaboratively building a single working software piece over a 10-day period.
* Like real-world scenarios, teams are not provided with all requirements upfront and are not given step-by-step instructions, fostering adaptability and problem-solving skills.
* Teams are expected to consistently deliver quality code, with typical iterations lasting only 20 minutes.
* "Do The Code Right" approach is emphasized, reminding teams to Design-Test (first!)-Code-Refactor every 20-30 minutes, promoting a cycle of continuous improvement and adherence to best practices.

Objectives

* Apply Agile principles in team projects to deliver high-quality, valuable software iteratively
* Implement Test-Driven Development (TDD) to ensure code quality and maintainability
* Demonstrate understanding of Object Oriented Principles and Practices in software design
* Utilize Design Patterns to solve common design problems efficiently
* Apply Lightweight Design principles to create flexible, maintainable software solutions.

Prerequisites

* Six months or more of programming experience in an object-oriented language

Outline

* Module 1: Introduction to TDD
  + Purpose and benefits of TDD
  + Motivation behind TDD: why it's crucial in modern software development
  + Overview of the TDD cycle: Red-Green-Refactor
* Module 2: Test Driven Development (TDD)
  + Principles and Techniques
  + TDD Metaphors
  + Benefits, Challenges and Limitations
  + Handling Requirements Change
  + Characteristics of good tests
  + Revisit Anti Patterns
* Module 3: Testable Designs (Mocks, Fakes and Stubs)
  + Creating testable Code, If you cannot test it what use is it?
  + Strategies for Testable Code
  + Test Unfriendly features
  + Interfaces are great!
  + Stubs, Fakes and Mocks
  + Mocks as Collaborators
  + Mocks and return values, void methods, frequency calls and ordering
* Module 4: OO Building Blocks
  + Classes and Objects
  + Operations and Methods
  + Instantiation of Objects
  + Inheritance
  + Overloading
  + Overriding
  + Interfaces
  + Abstract Classes
* Module 5: Encapsulation
  + Data Hiding
  + Type Hiding
  + Polymorphism
  + Associations
  + Dependency and Delegation
  + Aggregation and Composition
  + Coupling
  + Cohesion
  + Redundancy
  + SOLID & DRY
* Module 6: Lightweight Design
* First Principles
* When to Design in Agile
* User Stories
* Class Diagrams
* Review Checks
* Module 7: Commonality and Variance
  + Techniques for Translating from Requirements
  + Fundamentals of Commonality/Variability Analysis (CVA)
  + CRC Cards
  + How to Handle Variations as We Get New Requirements
  + Understanding and Using Factories
* Module 8: Complete Code
  + Doing the Simplest Thing Possible
  + Testable
  + Proper Encapsulation
  + Strong Cohesion
  + Correct Coupling
  + Readability
* Module 9: Delegation
  + Delegation and Why it is so Powerful
  + How Various Design Patterns Leverage Delegation
  + Adapter Pattern
  + Strategy Pattern
* Module 10: Refactoring
  + What is Refactoring
  + Why Refactor
  + Handling API Changes
  + Identifying Code Smells
  + Refactoring and Testing
* Module 11: Using Abstraction
  + Understanding the Template Method Pattern
  + Importance of Depending on Abstraction
  + Serializing Objects to XML
* Module 12: Delegation
  + Delegation and Why it is so Powerful
  + How Various Design Patterns Leverage Delegation
  + Adapter Pattern
  + Strategy Pattern
* Module 13: MVC and MVVM
  + Components and Responsibilities of Each Architecture
  + Benefits and Drawbacks of Each Architecture
* Module 14: Managing Access
  + The Proxy Pattern
  + Lazy Instantiation
  + Cross-cutting Concerns
* Module 15: Dynamic Responsibilities
  + Adding Flexible Functionality
  + The Decorator Pattern
  + The Observer Pattern
  + Writing a Custom Decorator