**CT 765 02**

**Course Objectives**:

1. Deliver adaptable software iterations and releases based on Agile methodologies
2. Minimize bugs and maximize productivity with Test-Driven Development and Unit Testing
3. Refractor existing code for easier maintenance and improved design
4. Achieve quality design by adopting established coding principles
5. Provide an illustration on real life Agile Implementation through a case study in Extreme Programming
6. Adopt best practices to successfully manage Agile projects
7. **Review of Traditional Approaches[4 hours]**
   1. Overview of Waterfall Model
   2. Overview of Spiral Model
   3. Limitation of Traditional Approaches

1. **Introduction to Agile Methodologies[4 hours]**
   1. Need of Agile Methodologies
   2. Objectives of Agile Methodologies
   3. Agile Implementations and Variants
   4. Introduction to the Agile Manifesto

1. **Planning an Agile Project[6 hours]**
   1. Establishing the Agile project
      1. Adopting the best practices of the Agile Manifesto
      2. Recognizing the structure of an Agile team
      3. Programmers
      4. Managers
      5. Customers
   2. Developing a Foundation with User Stories
      1. Eliciting application requirements
      2. Writing user stories
   3. Estimating and “The Planning Game”
      1. Defining an estimation unit
      2. Distinguishing between release and iteration
      3. Prioritizing and selecting user stories with the customer
      4. Projecting team velocity for releases and iterations

1. **Agile Iterations[5 hours]**
   1. Breaking user stories into tasks
      1. Recognizing a program's main purpose
      2. Prioritizing tasks for a cohesive design
      3. The Agile coding process
      4. Write Test, Write Code, Refactor
      5. Allocating time for a spike

1. **Test Driven Development[12 hours]**
   1. Design process with automated testing
      1. Introduction to Test Driven Development
      2. Writing a User Acceptance Test
      3. Compiling and Running tests
   2. Integrating Unit Testing
      1. Distinguishing between user tests and unit tests
      2. Developing effective test suites
      3. Achieving "green lights" through continuous testing
   3. Optimizing test-driven development
      1. Drafting a unit test that is simple, isolated and fast
      2. Isolating classes for effective testing
      3. Creating mock objects for testing
   4. Refactoring
      1. Code Duplication
      2. Renaming fields and methods
      3. Extracting methods and base classes
      4. Programming by intention

1. **Managing Agile Projects[4 hours]**
   1. Delivering the first release
   2. Planning the next release
   3. Adapting Agile to fit Development Methodology

1. **Extreme Programming[10 hours]**
   1. Core Principles and Practices
   2. Requirements and User Stories
   3. Release Planning
   4. Iteration Planning
   5. Customer Tests
   6. Small, Regular Releases
   7. Pair Programming
   8. Continuous Integration
   9. Collective Code Ownership
   10. Team Roles
   11. Case Study

**References**

1. Robert C. Martin, Agile Software Development, Principles, Patterns, and Practices, Prentice Hall (2002)
2. Andrew Hunt, David Thomas,The Pragmatic Programmer: From Journeyman to Master,1st Edition,  Addison-Wesley Professional  (1999)

**Evaluation Scheme:**  
The questions will cover all the chapters of the syllabus. The evaluation scheme will be as indicated in the table below:

|  |  |  |
| --- | --- | --- |
| **Chapters** | **Hours** | **Marks Distribution\*** |
| 1 | 4 | 7 |
| 2 | 4 | 7 |
| 3 | 6 | 12 |
| 4 | 5 | 7 |
| 5 | 12 | 22 |
| 6 | 4 | 7 |
| 7 | 10 | 18 |
| **Total** | **45** | **80** |

\*There could be a minor deviation in Marks distribution