Publication date : 01/29/2024

Course : Test Driven Development in C#

Practical course - 3d - 21h00 - Ref. TDN
Price : 1940 CHF E.T.

★★★★★ 5 / 5

This course covers individual, pair and team practices. It teaches emergent design, refactoring of existing code and good coverage of automatic testing. It explains automatic functional testing, carried out in collaboration with functional experts and testers.

Teaching objectives

At the end of the training, the participant will be able to:

- ✓ Write automatic tests.
- ✓ Test isolation.
- ✓ Test-driven development.
- ✓ Behavior-driven development.
- ✓ Refactoring under test.

Intended audience

This course is aimed at engineers or technical leaders, architects, managers and project leaders in software development.

Prerequisites

Practical object programming in C#.

Practical details

Hands-on work

Alternating conceptual sequences and practical work.

Course schedule

PARTICIPANTS

This course is aimed at engineers or technical leaders, architects, managers and project leaders in software development.

PREREQUISITES

Practical object programming in C#.

TRAINER QUALIFICATIONS

The experts leading the training are specialists in the covered subjects. They have been approved by our instructional teams for both their professional knowledge and their teaching ability, for each course they teach. They have at least five to ten years of experience in their field and hold (or have held) decision-making positions in companies.

ASSESSMENT TERMS

The trainer evaluates each participant's academic progress throughout the training using multiple choice, scenarios, hands-on work and more.
Participants also complete a placement test before and after the course to measure the skills they've developed.

1 Agility

- The manifesto for agile software development.
- Agile principles.
- The signatories of the agile manifesto.
- Not one, but many agile methods and practices.
- eXtreme Programming.
- Turn the test pyramid upside down.
- Agile testing framework.
- Integrated proof: deliver code and proof of operation.

2 Automatic tests

- Test typology. Automatic tests.
- Use of NUnit test library or equivalent.
- Arrangement Action Assertion.
- Principle of test isolation for reproducibility.
- Separating production code and tests. Purpose and contract.
- Create a test on production code.
- Name a unit test.
- Roofing as a development tool.

Hands-on work
Live coding and automatic test development exercises.

3 Test isolation

- Notions of Fake, Stub & Mock.
- Insulation of a component to be tested.
- Isolating an integration to be tested.
- Insulation tools.
- Practical extract & override.
- Use of Moq library or equivalent.

Hands-on work
Live coding and extract & override and mock exercises.

4 Test-driven development

- TDD test driven development.
- Baby step.
- Finding a path to functionality.
- Use assertions to code.

Hands-on work
Live coding and TDD exercise.

TEACHING AIDS AND TECHNICAL RESOURCES

• The main teaching aids and instructional methods used in the training are audiovisual aids, documentation and course material, hands-on application exercises and corrected exercises for practical training courses, case studies and coverage of real cases for training seminars.

• At the end of each course or seminar, ORSYS provides participants with a course evaluation questionnaire that is analysed by our instructional teams.

• A check-in sheet for each half-day of attendance is provided at the end of the training, along with a course completion certificate if the trainee attended the entire session.

TERMS AND DEADLINES

Registration must be completed 24 hours before the start of the training.

ACCESSIBILITY FOR PEOPLE WITH DISABILITIES

Do you need special accessibility accommodations? Contact Mrs. Fosse, Disability Manager, at psh-accueil@orsys.fr to review your request and its feasibility.

5 Behavior-driven development

- Communication problems. User history, acceptance criteria.
- Functional testing.
- ATDD Acceptance Test Driven Development.
- Specification by example. 3 amigos strategy for a better dialogue. Sharing understanding of the business domain.
- Specifications and recipes in one continuous process. BDD Behavior Driven Development.
- Features and Steps. The BDD cycle.
- Use the Specflow library or equivalent.
- Behavior-driven development.

Hands-on work
Live coding and BDD exercise.

6 Development in pairs

- Pilot & co-pilot style.
- Ping pong style.
- Mutual support.
- Constant negotiation.

Hands-on work
TDD development exercise in pairs.

7 Refactoring under test.

- Historical code. Technical debt. Analysis Paralysis.
- Code that's 2 years old should be better than code that's 2 weeks old.
- Stop generating even more technical debt.
- The Boy Scout principle.
- Smoke and characterization tests. Parallel testing. Notion of code smells.
- A reminder of the SOLID principles of object-oriented programming.
- Improve where you work first.
- Use of automatic refactoring with Resharper or equivalent.

Hands-on work
Live coding and refactoring exercise.

Dates and locations

REMOTE CLASS
2026 : 16 Mar., 8 June, 19 Oct.