assurity

BETTER SOFTWARE DEV

DESCRIPTION

Experience has shown that creating maintainable code is best done when guided by tests. To do this effectively, developers must know how to recognise code smells, apply strategies for refactoring, as well as write tests for legacy and new code bases.

The course format mixes teaching, hands-on exercises with real code and sharing of the practitioner's experiences to empower participants to deliver quality, working testable software.

By partnering with Industrial Logic, one of the world's leading providers of training for developers, we are able to offer world-class software development training to New Zealand companies.

"Industrial Logic eLearning has reached over 30 Google offices worldwide, which makes it one of the most successful and scalable trainings in engEDU" – Amanda Ball, eLearning Program Manager, Google

As part of this course, all participants receive a lifelong licence to the Industrial Logic eLearning materials, allowing them to revisit the course material at their own leisure any time. This includes access to the 'Code Critique' tool – a powerful tool that constantly analyses your code and activity in real time to allow you to fine-tune your work and your style as you go. It is like having a team of experienced developers sitting with you as you work, offering praise and suggestions for improvement.

LEARNING OUTCOMES

By the end of this course, attendees will have learned how to:

- Apply a micro-testing (unit testing) approach to software development
- Apply Test-driven Development (TDD) to both new and legacy code
- Use a range of powerful techniques for preventing and reducing technical debt
- Employ a selection of techniques for refactoring code
- Enhance agility through technical excellence and good design
- Approach code with simplicity in mind (i.e. the art of maximising the amount of work not done)

SUMMARY

This course teaches software developers techniques to improve how they design and develop software. The focus is on continuous attention to technical excellence, good design and how this enhances agility. It is about writing maintainable code over code that is easy to write.

COURSE LEVEL

+ DEVELOPED BY ASSURITY

DURATION

3-day course

COST

For prices, refer to **education.assurity.nz**

BOOK ONLINE AT

education.assurity.nz

BETTER SOFTWARE DEVELOPMENT

- Build flexible code that can "welcome changing requirements, even late in development" (see agiliemanifesto.org)
- Gain techniques for safely working with legacy code and bringing it under control

AGENDA

The course is delivered in six key modules:

Code Smells

Identification and classification of technical debt into a metaphor of Code Smells. Code Smells provide industry wisdom about how not to design software, going beyond vague programming principles by describing specific, recurring problems in software design.

Refactoring

Refactoring is software design hygiene; a process for safely improving the design of existing code. If you refactor regularly and at the right time, you will avoid taking on large quantities of technical debt. This modules covers tools and techniques for reducing technical debt and improving code quality in terms of:

- Readability
- Maintainability
- Conciseness, cohesion and coupling
- Overall coding excellence

Micro-testing

Programmers who want to move faster when writing microtests. A microtest (also known as a unit test) automatically verifies a single behaviour of a single object. If you want to spend less time debugging or manually testing and more time programming faster or making important design changes, you're ready for micro testing.

Test-Driven Development

What's the fastest way to write world-class code? Learn to microtest before you code. Test-Driven Development (TDD) is a rhythm, a design technique, a development process and a revolution, all in one. If you want to evolve software with a simple, clear design and exceedingly good test coverage, then TDD is for you.

Faking and Mocking

Software never exists in a vacuum; everything beyond the smallest script has to collaborate with other components. This brings new challenges, especially in a Test Driven world where often we want isolation and speed. To work effectively with collaborations, improve your tests (and your design!) you need solutions in the form of fake and mock objects.

Legacy Code

Everyone is sure they know what legacy is. And to tell the truth, they do know: We've all seen and worked with code that was hard on many levels:

- Hard to understand
- Hard to change
- Hard to find defects
- Hard to fix them when you find them
- Hard to know when you're doing the right thing

Here we look at the universal steps that help you start working with legacy code, finding ways into an risky or hard to change codebase and then opening it up to change, using the tools toolbox developed over the previous modules.

BETTER SOFTWARE DEVELOPMENT

BOOKING

Book and pay online at assurity.co.nz/education. We offer a 10% discount for early bird bookings made more than one month in advance. We offer discounts for groups of six or more.

PREREQUISITES

Attendees are expected to have experience in either:

• Java programming using Eclipse or IntelliJ IDEA

• C# programming using Visual Studio

The course is also available in C by request.

Note: Attendees must bring a laptop to the course with either Eclipse (v3.5 or later), IntelliJ IDEA (14.1.1 or later) or Visual Studio (2010 or later) installed. If using Visual Studio, the JetBrains ReSharper plug-in (v7.1 recommended) must also be installed. A free trial licence for ReSharper is available.