Fundamentals Of Object Programming

Modern OOD, Design By Contract, Reusability, AOP, MDA, Design Patterns, Software Failure, Generics

Object oriented techniques has now been applied to a large number of projects with a varying degree of success. With a better understanding of where the real benefits lie, object technology is being continuously refined and improved accordingly. This training course brings software developers - who already have a working knowledge of the basics of OO - up to speed with modern object-oriented design and the evolving OO programming techniques and enables them to discover how to use objects successfully. It answers the question: "What is happening in the OO world, beyond the fundamentals such as inheritance, encapsulation and polymorphism?" The goal of any OO project is to produce the best software for as few resources (time /money/devs) as possible. The software should evolve well in the future and adapt to changing needs. Parts of the software should be reusable on other projects. It should behave robustly when it encounters errors.

You will benefit from attending this course by gaining a clear understanding of the very latest object technology concepts, understanding its vocabulary and identifying how it fits into the broader picture of s/w engineering when describing how to approach software tasks.

Target Audience

Software engineers wishing to learn about the latest advances in object-oriented concepts

Prerequisites

Some previous OO software development and general application design experience.

Contents of One-Day Training Course

Object Modeling

Why do we need to model?
What benefits does it produce?
Object technology involves a series of simple concepts – why is it so difficult to really get right?

Design By Contract

Design By Contract concepts
Preconditions/postconditions/invariants

Object Discovery

How do I find the appropriate objects
Object discovery techniques
Is this one object, two objects, or none?

Modern OOD

Getting the class and instance structure right. The importance of typing and ADT.
Meyer's Software Construction Principles:
Linguistic Modular Units
Self-documentation
Uniform Access
Open-Closed
Single Choice

Reusability

How to reuse objects
Design for reuse
Managing reuse

Aspect Oriented Programming

What are aspects
Reuse through the callstack
Setting up the call environment with AOP

MDA

Concept of Model Driven Architecture
How to apply MDA

Design Patterns

The essence of a good project is not in its codebase but in its design
A good design is far more reusable than a good piece of source code
Design patterns are a technique of capturing design experience for later reuse by the original and other designers

Software Failure

Identifying failure points
Catching, reporting and reacting to failure
Importance of handling failure within OO

Generics

Type-independent programming
Generics in various languages

Agile Modeling

Problems with "heavy-weight" processes
AM is a family of light-weight processes, suitable for demands of fluid projects
Modeling the AM way

eXtreme Programming (XP)

XP Principles: pair programming, integrate & test every day, continuous feedback, get to running code ASAP, evolutionary design, managing communication

Latest OO Ideas

Orderly vs. Experimental Engineering
AntiPatterns
Internet time
Designing for continuous change

Case Study

Detailed case study showing how to apply advanced OO concepts to a project