

Design Patterns

Pattern Template, GoF Patterns, Applying Patterns, Advanced Patterns, Anti-Patterns, Pattern Extraction

[[Detailed examples](#)] Design patterns capture successful design experience for later reuse by the original/other designers. They capture solutions that have evolved over time, in a concise and easily applied fashion. Typically they are not the “first attempt” at solving a problem – but rather the result of an iterative design process by experienced designers (who have benefited from hard-learned lessons of previous projects). Teams are under tremendous pressure to produce higher quality software at lower cost. One option is to ship the work to cheap offshore development partners. A better option is to use smaller, but much higher skilled teams - who will

compete by working more effectively to build the software. Such advanced teams will be trained in design patterns and hence can aggressively leverage them to retain a productivity and quality advantage over lesser skilled competition. As software projects increase in scale, cost and complexity, and involve more inter- and intra-company relationships, there is a need to adopt techniques such as design patterns, to ensure use of best practices in design issues. The goals of this course are to look at a range of design patterns, to examine how to apply them in your own projects and explore how to create your own pattern catalog.

Contents of One-Day Training Course	
<p>Target Audience This course targets senior software engineers and architects who need to leverage design patterns correctly in their own projects.</p> <p>Prerequisites Attendees require good OO knowledge and plenty of development experience.</p>	<p>Overview of Design Patterns Simple and elegant solutions Applying the concept of generics from programming to software architecture Catalog of design patterns</p> <p>Defining a Design Pattern Documentation template Important GoF fields (intent, motivation, applicability, structure, participants, collaborations, consequences) Additional fields by others</p> <p>GoF Creational Patterns Abstract Factory, Builder, Factory Method, Prototype, Singleton</p> <p>GoF Structural Patterns Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy</p> <p>GoF Behavioral Patterns Chain Of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor</p> <p>Applying Patterns Applying patterns in your projects Implementing patterns in code Evolution of project over time</p> <p>Design Patterns and ... eXtreme Programming Aspect Oriented Programming (AOP) Agile Development Documentation</p> <p>Server Platform Patterns Patterns to satisfy competing demands</p> <p>Pooling, tuning, managing, sharing, distributing, extending, scheduling</p> <p>Concurrency/Network Patterns Wrapper, Component configurator, Interceptor, Extension interface, Reactor, Proactor, Async completion token, Acceptor-connector, *-locking, Active object, Monitor object, Half-sync/halfasync, leader-follower</p> <p>Enterprise Integration Patterns Gregorgrams Message exchange, channels, headers</p> <p>Security Patterns How to correctly design security features into your software systems Secure channel, Session, Role, Checkpoint, Single access point, Full/limited views</p> <p>Anti-Patterns Anti-patterns “let you zero in on the development detonators, architectural tripwires and personality booby-traps that can spell doom for your project” (Brown)</p> <p>Pattern Extraction Finding patterns in your own projects Effort involved in extraction Optimizing and generalizing patterns</p> <p>Custom Pattern Catalog How development teams can build up a catalog of patterns for their own needs</p> <p>Project Using Patterns An advanced project whose architecture uses an assortment of design patterns</p>