C# and Design Patterns (code CSDP)

Part I: Structural Patterns and Infrastructure

Structuring a C# Application

- Whole Part Pattern
- Model View Controller (MVC)
- Presentation Abstraction Control (PAC)

Interfacing in C#

- Interfaces and abstract classes
- Provides and requires interfaces
- Contracts and service-oriented programming
- Contracts in C# (the where clausule)

Defining Structure

- Inheritance
- Aggregation and Composition
- Delegation: stateful and stateless variants

Short Overview of UML 2.0 Diagrams

- Class diagrams
- Component diagrams
- Assembly and connector components
- Sequence diagrams and statecharts

Part II: GOF Patterns in C#

Overview

- Background to design patterns
- GOF pattern classification
- 20-80% rule: essential and non-essential patterns
- Advantages of using design patterns

Creational Design patterns

- Applicability of creational patterns to C#
- The Builder pattern
- Factory Method and Abstract Factory patterns
- Prototype pattern
- Using .NET Reflection for factories: Activator

Structural Design Patterns

- Composite pattern and nested objects
- Proxy pattern
- Decorator pattern
- Class and object Adapter
- Bridge pattern

Behavioural Patterns I

- Command pattern
- Strategy pattern
- Using Delegates for creating algorithms
- Using interfaces and inheritance for Template Method pattern
- Visitor pattern and service extension
- Iterator pattern and its implementation in .NET

Behavioural Patterns II

- Observer (Publisher-Subscriber) pattern
- Mediator pattern and change control
- Using events and Delegates as an alternative to Observer

Other Behavioural Patterns

- State pattern
- Role pattern
- Propagator pattern

Part III: Patterns using C# Generics

Policy-based Design in C#

- Implementing contracts with interfaces
- Combining behaviour and structure: interfaces and inheritance
- Moving GOF patterns to their generic forms

Generic Patterns

- Strategy pattern
- Bridge pattern
- Command pattern

Part IV: Patterns and C# Extensions

Regular Expressions

- What is a regular expression?
- Matching
- Compiled regular expressions
- Regex options

Lambda Expressions

- Using lambda expressions instead of delegates
- Expression trees
- Generic lambda expressions
- Anonymous methods

Serialisation

- Serialisation engines
- Explicit and implicit serialization
- Binary and XML serialization
- Integration with Builder and Visitor patterns

Multi-threaded patterns

- Master-slave pattern
- Producer-Consumer pattern

- Thread pooling

Examples

- Mini 2D CAD application
- Extended Producer-Consumer
- Clock Publisher-Subscriber

Your Trainer

Daniel J. Duffy started the company Datasim in 1987 to promote C++ as a new object-oriented language for developing applications in the roles of developer, architect and requirements analyst to help clients design and analyse software systems for Computer Aided Design (CAD), process control and hardware-software systems, logistics, holography (optical technology) and computational finance. He used a combination of top-down functional decomposition and bottom-up object-oriented programming techniques to create stable and extendible applications (for a discussion, see Duffy 2004 where we have grouped applications into domain categories). Previous to Datasim he worked on engineering applications in oil and gas and semiconductor industries using a range of numerical methods (for example, the finite element method (FEM)) on mainframe and mini-computers. Daniel Duffy has BA (Mod), MSc and PhD degrees in pure and applied mathematics and has been active in promoting partial differential equation (PDE) and finite difference methods (FDM) for applications in computational finance. He was responsible for the introduction of the Fractional Step (Soviet Splitting) method and the Alternating Direction Explicit (ADE) method in computational finance. He is also the originator of the exponential fitting method for time-dependent partial differential equations.

He is also the originator of two very popular C++ online courses (both C++98 and C++11/14) on www.quantnet.com in cooperation with Quantnet LLC and Baruch College (CUNY), NYC. He also trains developers and designers around the world. He can be contacted dduffy@datasim.nl for queries, information and course venues, in-company course and course dates