Programming Generic and Parallel Design Patterns in C++ (code CPP-PDP)

Course Contents

Part 1: Generic Design Patterns

A Review of the GOF (Gamma) Patterns

- Foundations (polymorphism and delegation)
- Creational, structural and behavioural patterns
- Consequences of using the object-oriented model
- Action points for using new design models

Designing Classes: Fundamental Techniques

- Object Connection architecture (OCA)
- Interface Connection architecture (ICA)
- Plug and socket architecture (PSA)
- Which architecture to choose and when?

Introduction to generic Design Patterns

- What is a generic design pattern?
- Relationship with connection architectures
- Modelling structure and behaviour
- Combing object and generic architectures

Generic Design Building Blocks

- Class traits
- Policy-based design and C++ templates
- Template Template Parameters (TTP)
- Curiously Recurring Template Pattern (CRTP)
- Using Design Building Blocks
- TTP for aggregation and composition
- Implementation of static polymorphism with CRTP
- UML object structures using OOP and GP
- Review: efficiency, reusability, reliability and maintainability

Part 2: Design Patterns in Boost

Overview of Boost Libraries

- Boost library categories
- Boost libraries and design patterns
- Boost libraries that directly support design patterns
- Designing with Boost libraries

Flyweight (Boost) Pattern

- Creating and managing shared objects (intrinsic/extrinsic data)
- Key-value flyweights
- Flyweight and composites
- Flyweight factories
- Multi-threading

Boost Signals and Slots Library

- Creating multicast callbacks for functions and function objects
- Triggering and handling events: loose coupling
- Connecting slots and slot grouping
- Signals with arguments
- Multi-threaded Signals library

The GOF Observer Pattern

- What is not right with the GOF Observer?
- Using Signals and Slots to implement the Observer
- Multiple publishers, multiple publishers
- Application: Generic Model View Controller

The generic Command Pattern

- A review of the GOF Command pattern: what are the issues?
- Using Boost.Function to create object-level Command patterns
- Boost.Regex as an interpreter for user requests
- Composite commands

Generic Strategy Pattern

- Template member function solution
- Using generic function objects
- Where does Boost.Function fit in?
- Advantages of using generic Strategy

Part 3: Multi-Threaded Patterns and their Implementation

Overview of Parallel Software Design Process

- Finding Concurrency: task and data decomposition
- Algorithm structure design
- Supporting parallel design patterns
- Implementation mechanisms

Multithreading Basics

- Thread class and lifecycle
- Shared and private data
- Mutual exclusion
- Barriers and fences
- Thread synchronization and notification

OpenMP

- Core concepts and runtime library
- Structured blocks and directive formats
- Worksharing Synchronisation; scheduling
- Parallel design patterns in OpenMP

Boost.Thread

- Thread class and thread start options
- Mutex and locking
- Wait and notify
- Condition variables
- Thread groups
- Parallel design patterns in Boost Thread

Algorithm Structure Patterns

- Task parallelism
- Divide and Conquer
- Geometric Decomposition
- Producer-Consumer

Supporting Structures Patterns

- Master/Worker
- Single Program Multiple Data (SPMD)
- Fork/Join
- Loop Parallelism

Troubleshooting

- Sequential consistency
- Removing data dependencies
- Race conditions
- Deadlock and livelock

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