Modern Multiparadigm Software Architectures and Design Patterns
with Examples and Applications in C++, C# and Python Volume I

Datasim Press (planned publication date December 2023)

Daniel J. Duffy dduffy@datasim.nl and Harold Kasperink harold.kasperink@enjoytocode.com

Summary

The main goal of this book is to introduce a repeatable and structured approach to modern software design patterns and best practices. In contrast to most books on computing we address the full software lifecycle from problem description, system design, next generation multiparadigm design patterns through to implementation in C++, Python and C#. The focus is very much hands-on and the examples and applications are based on real-life projects and pedagogically robust examples. We take an incremental, step-by-step approach by first taking motivational examples in Python and we can then generalise and subsume them in larger applications in C++ and C#. To this end, the first five chapters discuss most of the object-oriented, functional and generic programming language features that we need in later chapters.

Chapters six to fourteen build on the foundations of the first five chapters which were concerned with implementation details and solving problems. We now address and mitigate some of the major maintenance issues that challenge software projects and to this end we propose a process by first analysing the problem to be solved, proceeding to a system design of the problem and then finally a detailed design and implementation in one or more languages. These chapters include original work as well as design techniques from the early years of the object-oriented revolution.

The remaining chapters and appendices in the book discuss a number of applications of the design patterns and language features that we introduced in previous chapters. We examine several concrete examples and we show how to design and implement them in a given programming language. Studying these applications will help you apply designs to your own use cases, thanks to the fact they are instances of one or more domain architecture categories.

In short, we have made some effort to produce a unique body of work that describes the life of software products from idea to running code!

Main Topics

The contents of the book discusses the skills to be learned in order to write well-designed and maintainable software systems. The topics are chosen in such a way to reflect these goals. In other words, we adopt proven best practices in software development projects. To this end, we structure the chapters based on the following attention areas:

- Preparation: object-oriented, functional and generic programming styles in C++, Python and C#.
- Object-oriented design principles.
- Beyond object-orientation: requirements analysis, system decomposition and Domain Architectures.
- GOF (Gamma) design patterns, examples, critique and generalisations using C++ Concepts.
- Several chapters on applications, from problem description to running code.

In short, this book uses a defined and repeatable process that integrates and subsumes several best design and programming practices, including original extensions and improvements from the authors. These can be seen as one of the major features of the book.

Special Features in this Book

This book represents the authors' experience in industry, academia and professional training institutions in the roles of programmer, designer, requirements analyst, teacher/coach and CEO. Based on both of our technical and business backgrounds we have attempted to write the book in a such a way that we do things right as well as doing the right things!

- The full software process from problem description to running Python, C++ and C# programs.
- The first book in our opinion to discuss software development in this way (based on more than combined 60 years industrial experience of the authors).
- Incremental engineering approach: start with simple examples and progress to more complex applications.
- Using domain architectures to design reusable and robust applications.
- Reengineering and major overhaul of the famous GOF (Gamma) design patterns.
- Language-independent design with defined and repeatable mappings to Python, C++ and C#.
- Programming language-integration, 'use the best of many worlds.'
- Deeper understanding of modern programming styles.
- Guidelines, best practices, prototypical models that can be used to bootstrap your software projects and applications.
- Each chapter has well thought-out hands-on coding exercises and projects to consolidate your skills. In short, this book is a resource for designers and developers who wish to write maintainable applications in C++, Python and C#.

For whom is this Book?

This book is for a wide range of novice and experienced programmers and designers. We assume some knowledge of at least one object-oriented programming language. We introduce enough syntax and language features of C++, Python and C# in order to benefit from the designs that we introduce. To this end, we think that this book can be of benefit to the following reader groups:

- Self-taught novice Python developers who wish to write maintainable code.
- C# and C++ developers who are interested in learning Python essentials.
- Object-oriented developers who wish to learn functional and generic programming.
- Software designers and architects who need to write complex and maintainable software systems.
- Software specialists who wish to learn state-of-the-art system and design patterns.
- (Data) scientists who write scripts and who wish to create software products.
- Researchers and scientists who write experimental code and who wish to build reusable and extendible software products.

In short, this book is also a valuable reference for a wide range of developers.

Ongoing/Continuing Support and Activities

Writing a book is only the beginning of a journey to continue your education and extend your skills! We offer a comprehensive suite of courses on Python, C++ and C# (at both beginner and advanced levels) as well as pure, applied and numerical mathematics and applications in engineering and computational finance:

Online courses

Regular courses

Quantnet

Python courses can found here:
Distance-learning-object-oriented-and-functional-programming-in-python-language-libraries-and-modern-design-patterns
Distance-learning-applied-numerical-methods-with-python-and-python-libraries

If you have any inquiries on the applicability, prerequisites, company/student pricing and other queries, please do not hesitate to contact us Daniel J. Duffy dduffy@datasim.nl and Harold Kasperink harold.kasperink@enjoytocode.com

We also coach and mentor developers and designers at all levels as well as offering in-company, site licenses, online and specialised customised courses.

Chapters in the Book

Each chapter has a good mix of analysis, design, code, syntax, exercises and quizzes.

Volume II (planned date Q2 2024)
A second book (Volume II) will discuss more extended applications and how to write code that simultaneously interoperates with C++, Python and C# at run-time. To this end, we discuss Cython, Boost Python, pybind11, Python .NET and C++/CLI interoperable applications.

Volume II builds on the functionality and design knowledge from Volume I. The rationale for this book is based on a number of assumptions:
- There is no 'universal' programming language that solves all problems; in general, programming languages were designed with a given objective in mind.
- We wish to reuse proven and tested legacy code that has been written in various programming languages. The most effective and proven resolution is to create wrappers for this code.

- Many developers are comfortable with at most one or two languages. It is not feasible for them to learn new languages just to be able to use a library. And they may not be inclined to!
- Choosing the appropriate mix of languages will result in better quality code at lower costs.

Provisional topics in Volume II are:
- A complete discussion of C++, C# and Python interoperability.
- Creating motivational examples.
- Integration with software libraries.
- Parallel and distributed design patterns.
- Applications: Tracking, computational finance, Statistical Learning, databases, numerics.

In short, the focus is on applying modern multiparadigm methods and libraries to producing realistic applications that readers can use as a baseline for their own work.

The Authors
Daniel J. Duffy is founder and owner of Datasim Education BV (founded 1989), which was one of the first pioneering companies to promote C++, object-oriented programming, system design and training in a range of industries such as telecommunications, Computer-Aided Design (CAD), holography (optical technology), process control, manufacturing systems and computational finance. In particular, these applications were designed using a combination of domain architectures (Duffy (2004)) and the famous design patterns that became popular in the 1990s.

Daniel Duffy is an internationally known author, trainer and numerical analyst. He has written more than ten books on programming, mathematics and computational finance. He has a PhD in mathematics (PDE/FDM) from the University of Dublin (Trinity College), much of which he uses in daily work and algorithms.

He is also the originator of two major and market leader C++ courses in cooperation with Baruch College, NYC (ranked #1 MFE school in 2023) and Quantnet Inc. (www.quantnet.com):
Quantnet C++
Quantnet Advanced C++

These courses are taken by many professionals and MSc/MEF students in forty countries and on five continents. Student testimonials:
C++ Online Programming Cert Testimonials

Harold Kasperink is CEO of NCIM, a software system integrator focusing on building complex systems for various clients. In 1997 he followed his first Design Pattern course with Daniel Duffy and since then he has worked on object-oriented system development, design patterns and system architecture in C++, C#, Java and Python. Over the past 25 years he has been involved in multiple projects covering simulation, real-time systems, communications, web and distributed systems. He has given various talks and training, as well as guiding developers with programming and designing software.

Harold is currently researching high performance computing and interoperability of distributed systems at both the network level and integration between programming languages. He holds a BSc degree in Computer Science and a MBA from the University of Liverpool.