

PREFACE

Why this book?

This text is a **handbook** for students and engineers. It introduces them to the creation and implementation of space-related models, which are models based on geometrical points that are modeled by a *MyPoint* class. Based on a variety of examples formulated during my ten years of teaching and research, the book takes a learning-by-doing and problem-oriented approach to teach the skills needed to build spatial models and then to write a computer program for the model. These procedural skills are rarely taught at universities and many engineers struggle to transfer a model into a computer program. The purpose of this book is to fill this gap. It moves from simple to more complex applications covering various important topics in the sequence: dynamic matrix processing, 2D and 3D graphics, databases, Java applets and parallel computing.

This book is designed to **combine theory and practice** of building models with objects and patterns for spatial applications in natural sciences from a practical point of view by **integrating** object orientation and patterns into space-related model building. All example programs in this book except the Java applets use a **file as the main building block** (*MyFile* as the base class) to create applications containing a parent/multiple child object system as the underlying design structure. Here a file can be of the following type:

- Data text (ASCII) files,
- Graphics or image files,
- Database files, or
- Executable files with “exe” as file extension.

It will be demonstrated in part II of this book how these different file types can be processed by building on **just one base class** (*MyFile*) and retain the general design and implementation of a variety of window programs. The main reason for this is the seamless application of design patterns (mainly the builder, factory method and prototype patterns). For instance, the first example program can simultaneously open and read data text and graphics files. By extending this approach to database and executable files, a simple and powerful design method is at hand to build other wanted applications.

All example programs in part II of this book are **complete extendable medium-sized** applications that address real needs in natural sciences. These examples are more than just scattered pieces of computer code without relevance to the reader who is often confused by the huge amount of material (e.g. most of the Deitel computer books have at least 1000 pages, <http://www.deitel.com>) and dissatisfied by tiny example programs. Instead, this book tackles a serious problem for many readers today: **how to find relevant information** and parts for building models and how to assemble them? This text provides a **solution** by offering a wisely selected and versatile collection of important applications, and by guiding readers through this material using their sequence diagrams. Sequence diagrams are similar to flow charts and show the flow of control and message passing between objects of an application in a timely sequence. They are very useful for explaining in a simple way in which a more complex application works. These explanations in part II of this book are based on “walking through” the steps of a sequence diagram and describing all object interactions and their operations, which are similar to functions and subroutines in Fortran. This also helps a reader to keep the overview and control over an application.

When thinking about the need for a simple, clean and functional method for average model builders, it has become obvious that the present standard for object-oriented system development, the **unified modeling language** (UML), is too complex and confusing for many people who are overwhelmed by the huge amount of UML concepts and diagrams. They do not know how to find what is important for them, and how to build their own models. This is one reason why the UML predecessors were included in this book. They are easier to understand and apply. Other reasons are to help to comprehend the UML because it is a merger of older modeling techniques, and to help to select from the UML what an average model builder just needs in terms of a simple, clean and functional approach (about 10% of the total UML). Simply speaking, two parts are required. Firstly, you have to select an overall architecture for your model and secondly, you need to have a working plan to guide and tell you how to build this model with its architecture. Both parts are provided in this text. The working plan consists of sequence diagrams mentioned before. The overall architecture is a **three-tier architecture** with a graphical user interface as top layer, a processor as middle layer and a data manager as bottom layer.

In this way, this book wants to assist students, engineers and other professionals, who have either no or only little knowledge about OO techniques and patterns, in their desire to create their own and/or alter existing models and computer codes. In doing so, any writer of such a text and any model builder face the same basic conflict: **complexity versus simplicity**. A high-quality model is a compromise of both extremes and its creation requires a lot of skill and experience. In the author's opinion, **learning-by-doing** is the most effective method of knowledge acquisition for building successful models. Therefore, it is strongly recommended to start practicing as soon as possible. Further, this book tries to help the reader by using a **step-by-step** writing approach moving from simple to more complex models and

examples. And where suitable, the **history** of covered material is described so that the reader can better comprehend the background and reasons why things happened as they evolved in time. This helps to build a deeper understanding.

Generally, creating and verifying models is a fundamental part of any scientific discipline that comprehends life and uses models in order to reach pre-defined goals. **Models** are representations of something real that can be explored and manipulated to get a deeper insight into its nature. Models can be transferred into products or services that satisfy human needs. Before applying a model, its correctness needs to be tested by sufficient experiments and found to be true within an acceptable fault tolerance (Popper 2002). To make the task easier and less cost intensive, modeling concepts and tools should be appropriate and efficient to do the job. They should be allowed to represent reality as good as possible and narrow the gap between the model and its real counterpart. Object-oriented (OO) methods and patterns belong to the most promising techniques for establishing a more ideal building approach for models.

Finally, this preface has to conclude with a remark about a general discussion regarding model building. Is this activity more an **art**, or does it belong more to the natural sciences with a pure interest in functionality? As in real life, there is no clear answer to this question. Most likely it is a good mixture of both. As a successful car model or a favorite Web page combines functionality with beauty, a high-quality model can also look and feel good!

This book is divided into two parts. Part I, **Object-Oriented Methodology**, consists of three chapters. Chapter 1 introduces the topic with its historical background and key concepts followed by a breadth-first example about OO methodology. Chapter 2 presents different OO modeling techniques including the “unified modeling language” (UML) and patterns. Chapter 3 presents model building with classes, objects, connections, attributes and operations.

Part II, **Model Building with Objects and Patterns**, is made up of seven chapters and applies the contents of part I to spatial problems in natural sciences. All these chapters are **organized in the same pattern**: after an introduction to the fundamentals the model to be created is described by the underlying requirements followed by an OO analysis and design. Then the model’s implementation, applied patterns and testing are explained. In more detail, part II introduces the following applications. Chapter 4 deals with a dynamic matrix processor in Visual Basic, chapter 5 with a 2D dynamic data plotter in Visual Basic, and chapter 6 with 3D visualization and animation in Visual Basic. Chapter 7 is about updating legacy programs written in e.g. C/C++, Fortran or Pascal by integrating them into an interactive window shell, thus retaining the investments in these programs. Chapter 8 introduces the reader to a database application for digitizing digital images. Chapter 9 describes another database application: a city map Java applet with a road finder. At the end, chapter 10 introduces parallel computing with Java threads (also called “multi-threading”) to determine orbits of earth satellites.

Spatial Modeling in Natural Sciences and Engineering
Software Development and Implementation

Friedrich, J.

2004, XV, 305 p. 36 illus. in color., Hardcover

ISBN: 978-3-540-20877-8