

Preface to the Second Edition

The first edition of this book was used as a textbook in several universities around the world. We have received comments and suggestions on how to improve the book. As the first edition was published in 2006, it is fitting that after 9 years there are many technological changes and advances that must be presented. For one thing, UML, which is at the heart of the approach used in the first edition, is in version 2. The new version of UML is more complete; it is enriched with the object constraint language (OCL) to better express constraints on modeling elements. This second edition has a dedicated chapter about OCL.

The framework-based software development is one of the most important approaches nowadays; it is powerful and enormously facilitates the process of software development. There are a few such frameworks such as Zend Framework (<http://www.framework.zend.com>) and Yii (<http://www.yiiframework.com>) to name a couple. At the center of this framework-based approach is the model-view-controller (MVC) design pattern. Therefore, we have dedicated a special section to the presentation of the MVC pattern.

In the Epilogue of the first edition we had predicted that the model-driven architecture (MDA) approach would grow and play a relevant role in the software development industry. Our prediction was right. MDA is becoming more and more prevalent in the software industry. Several MDA compliant tools have been present in the market and its share in the software development industry is growing constantly. One cannot write a serious book on software development without mentioning this approach. In this book, we have selected to introduce only two of the MDA-based tools—the virtual enterprise (<http://www.intelliun.com>) and Oliva Nova (<http://www.integranova.com>). Two new chapters present some details of these important technologies.

To write a book about software engineering is very challenging as technologies change continuously and books in this domain usually do not have a long shelf life. We paid special attention to not get into the details of the implementation of a particular technology but rather in presenting the modeling part of the problem. Models are abstract and technology independent. Therefore, models have a long shelf life.

Chapter 8 includes exercises and questions on conceptual modeling. The corresponding answers are given in Chap. 16 at the end of the book. The goal is to make readers aware that modeling is a complex task and the essence of the object-oriented paradigm must be understood before starting to model.

The book can be useful to undergraduate and graduate students and researchers who have an interest on modeling complex systems.

A book is never the result of the author's work only, and this book does not make exception from this general rule. Several are they who deserve credit for their objective and unselfish help that made this book better. We would like to express our gratitude to all of them for their criticism and suggestions that contributed to improve the quality of our work.

Preface to the First Edition

This book is an effort to bring the application of new technologies into the domain of agriculture. Historically, agriculture has been relatively behind the industrial sector in using and adapting to new technologies. One of the reasons for the technological gap between industrial and agricultural sectors could be the modest amounts of investments made in the field of agriculture compared to the impressive numbers and efforts the industrial sector invests in new technologies. Another reason could be the relatively slow process of updating the student's curriculum with new technologies in university departments that prepare our future specialists in the field of agriculture.

With this book, we would like to narrow the technological gap existing between agriculture and the industrial sector in the field of software engineering. We have tried to apply modern software engineering techniques in modeling agricultural systems. Our approach is based on using the object-oriented paradigm and the unified modeling language (UML) to analyze, design, and implement agricultural systems.

Object-oriented has been the mainstream approach in the software industry for the last decade, but its acceptance by the community of agricultural modelers has been rather modest. There are a great number of researchers who still feel comfortable using traditional programming techniques in developing new models for agricultural systems. Although the use of the object-oriented paradigm will certainly not make the simulation models predict any better, it will surely increase the productivity, flexibility, reuse, and quality of the software produced.

The success of the object-oriented approach is mostly due to the ability of this paradigm to create adequate abstractions. Abstraction is an effective way to manage complexity as it allows for focusing on important, essential, or distinguishing aspects of a problem under study. Object-oriented is the best approach to mimic real world phenomena. Entities or concepts in a problem domain are conceived as objects provided with data and behavior to play a well-defined role. Objects can represent anything in the real world, such as a person, a car, or a physiological process occurring in a plant. The use of objects enormously facilitates the process of conceptual modeling, which can be defined as the process of organizing our

knowledge of an application domain into orderings of abstractions to obtain a better understanding of the problem under study. Conceptual modeling makes heavy use of abstraction, and the object-oriented approach, unlike other programming paradigms, provides direct support for the principle of abstraction.

Currently, UML is an industry standard for visualizing, specifying, constructing, and documenting all the steps of the software development. UML allows for presenting different views of the system under study using several diagrams focusing on the static and the dynamic aspects of the system. UML can be used in combination with a traditional programming environment, but its power and elegance fits naturally with the object-oriented approach.

One of the most beneficial advantages of UML is its ability to design a platform independent model (PIM) that is a representation of the model using a high level of abstraction. Details of the model can be expressed clearly and precisely in UML as it does not use any particular formalism. The intellectual capital invested in the model is insulated from changes in the implementation technologies.

A platform specific model (PSM) is developed by mapping a PIM to a particular computer platform and a specific programming environment. A mapping process allows the transformation of the abstract PIM into a particular PSM. This two-layer concept, a PIM and the corresponding PSM, keeps the business logic apart from the implementation technologies. Experience shows that the business logic has a much longer life than the implementation technologies. Changes and evolution of the implementation technologies should not have any impact on the business model.

The book is divided into two parts. Part one presents the basic concepts of the object-oriented approach, their UML notations, and an introduction to the UML modeling artifacts. Several diagrams are used to present the static and dynamic aspects of the system. There are an ample number of examples taken from the agriculture domain to explain the object-oriented concepts and the UML modeling artifacts. In this part of the book, a short introduction to design patterns explains the need for using proven solutions to agricultural problems.

Part two deals with applying the object-oriented concepts and UML modeling artifacts for solving practical and real problems. Detailed analysis are provided to show how to depict objects in a real problem domain and how to use advanced software engineering techniques to construct better software. Examples are illustrated using the Java programming language.

The book aims to present modeling issues a designer has to deal with during the process of developing software applications in agriculture. Although the Java programming language is used to illustrate code implementation, this book is not intended to teach how to program in Java. For this topic, we would recommend the reader to look for more specialized books. There is a chapter in this book that introduces the reader to some of the design patterns that we have used in agricultural applications. In no way do we pretend to have covered entirely the subject of how to use design patterns in software development. For an advanced and full presentation of the design patterns, we strongly suggest the reader to consider the well-known book *Design Patterns Elements of Reusable Object-Oriented Software* [41].

Our approach is based on the rational unified process (RUP) methodology, although it does not rigorously follow this methodology. Our focus is on presenting modeling issues during the analysis and design of agricultural systems. For a more detailed and advanced approach to RUP, the reader needs to consult more specialized books.

What makes our book of unique value? Well, we have assembled in a comprehensive way a wide range of advanced software engineering techniques that will allow the reader to understand and apply these techniques in developing software applications in agriculture and related sciences. Agricultural systems tend to be more abstract than business systems. Everyone has a good understanding of how to use an ATM (automated teller machine). The use of an ATM is a classic example, used in many publications, to explain what an object is and how to build a UML diagram. The process of photosynthesis or the interaction of a plant with the surrounding environment, just to name a few typical agricultural examples, is less known to a large number of readers. Modeling a plant as an object provided with data and behavior may not be as straightforward as modeling an ATM. Therefore, the book aims to provide examples and solutions to modeling agricultural systems using the object-oriented paradigm and the UML.

The book is intended to be of use to anyone who is involved in software development projects in agriculture: managers, team leaders, developers, and modelers of agricultural systems. Developing a successful software project in agriculture requires a multidisciplinary team: specialists from different fields with different scientific backgrounds. It is crucial to the success of the project that specialists involved in the project have a common language that everybody understands. We find that UML is an excellent tool for analyzing, designing, and documenting software projects. Models can be developed visually and using plain English (and any other language for that matter) and can be understood by programmers and non-programmers alike. Thus, collaboration between these groups is substantially improved by increasing the number of specialists directly involved in the process of software design and implementation.

The book was written having always in mind the important number of specialists who still develop agricultural models using traditional approaches. There are ample step-by-step examples in this book that show how to depict concepts from a problem domain and represent them using objects and UML diagrams. We hope this book will be useful to these researchers and help them make a soft switch to the object-oriented paradigm. We hope readers will find this book of interest.

Tirana, Albania
Gainesville, FL, USA
May 2006

Petra J. Papajorgji
Panos M. Pardalos

Software Engineering Techniques Applied to
Agricultural Systems
An Object-Oriented and UML Approach
Papajorgji, P.J.; Pardalos, P.
2014, XVII, 301 p. 239 illus., 89 illus. in color.,
Hardcover
ISBN: 978-1-4899-7462-4