Preface

This book provides a set of readings on the state-of-the-art and the state-of-the-practice of the Unified Modeling Language (UML) and the Unified Process (UP).

UML is a language for specifying, visualizing, constructing and documenting software-intensive systems. It is a unifier of proven software modeling languages that appeared in the early 1990s. UML incorporates the object-oriented community’s consensus on core modeling concepts and includes an additional expressiveness to handle problems that previous languages (Booch, OMT, OOSE) did not fully address. It emerged in response to a call for a standard object-oriented and design method by the Object Management Group (OMG) in 1997. Currently, the OMG/UML standard is version 1.4 and the evolution of UML will result in version 2.0.

The UML notation includes diagrams that provide multiple perspectives of the system under analysis or development. It is layered architecturally and organized by packages. The model’s elements are defined in terms of their abstract syntax, well-formed rules (using the Object Constraint Language and precise text) and precise text. The formalization of UML is still an open problem. Many works have been done to formalize parts of the language and it is difficult to see how to integrate the works in order to define a formal semantics for UML.

Although UML does not prescribe any particular development process, various companies are working on processes to provide advice on the use of UML in the software development life cycle.

The OMG presented the “Software Process Engineering Metamodel” (SPEM) as a standard in November 2001. This metamodel is used to describe a concrete software development process or a family of related software development processes that use the UML notation. SPEM has a four-layered architecture of modeling for describing performing process, process model, process metamodel and MetaObject facility. Several processes fit SPEM. The most popular of these is the Rational Unified Process (RUP), developed and marketed by Rational Software. It is a software development process based on UML that is use-driven, architecture-centered, iterative and risk-driven. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. RUP is organized around four phases: inception, elaboration, construction and transition and core workflows: requirements, capture, analysis, design, implementation and test. Various industry sectors around the world use RUP in different applications: telecommunications, transportation, aerospace, defense, manufacturing and financial services.

UML and UP are having a significant impact on the software development industry. However, numerous practical difficulties have been detected with their use. As a result, they must evolve further by looking toward extension practitioners and re-
searchers to address specific concerns, then incorporate their feedback. In this direction, many innovations are concerned with the development of new theories and practices that are required to clarify and to make precise its semantics and reasons underlying properties of UML models. New theories and practices also transform software modeling and designs into code and enable object-oriented visual modeling tool interoperability.

There are still important issues to be solved in a satisfactory way. Techniques that currently exist in UML CASE tools provide little support for validating models in the design stages and are insufficient for completed, automated code generation. Little work has been done in order to investigate the use of well-proven and accepted requirements, techniques and models for the business and requirements of UP models. Some problems have been detected in the process of MDA methods that require flexible code generation mechanisms. Modeling of a performance-oriented, parallel and distributed application in UML is still an open problem.

This book collects insightful contributions from both industry and academia, illustrating how UML can be used, identifying open problems with UML and UP and suggesting solutions. The different chapters present perspectives on the UML and UP with respect to the following topics:

- Extensions and restrictions of UML and UP,
- Business process and modeling,
- Semantics,
- Mapping of UML models to frameworks, databases, formal languages and programming languages,
- Software components,
- Profiles,
- Security.

As an edited collection, this book should be of interest to practitioners, researchers and instructors of UML and UP.

**ORGANIZATION OF THE BOOK**

The book is organized into 19 chapters. A brief description of each chapter follows.

Chapter 1 provides a critical look at UML. The evaluation is done using a general framework for understanding the quality of models and modeling languages in the information systems. The authors argue that although being an improvement over its predecessors, UML still has many limitations and deficiencies related to both the expressiveness and comprehensiveness of the language.

Chapter 2 describes a generic framework for tailoring general purpose and model-based methodologies in order to deliver domain-specific models and to ensure utilization of existing knowledge possessed within the actual domain. By applying the tailoring framework, a domain-specific reference model is presented that consists of UML profiles, reusable models and patterns.

Chapter 3 analyzes the possibility of applying UML to design an inter-firm, online business model using UML. The proposed framework comprises such principal elements as value, business players and relationships among players, with each speci-
fied in terms of representative attributes and incorporating related notations. The business model is then visualized by value and structure diagrams.

Chapter 4 describes an approach for specifying business components. The authors propose a general and layered structure of software contracts for business components and show the shortcomings of common specification approaches. They introduce a formal notation for the specification of business components based on temporal extensions of the OCL.

Chapter 5 evaluates UML’s support for reuse using a modeling framework based on semiotic theory. This chapter explores the nature of modeling abstractions that could support the negotiation between stakeholders. The authors analyze two scenarios: one based on composable, functional abstractions and the other using structural abstractions as the basis for component compositions.

Chapter 6 describes a strategy that is based on natural, language-oriented requirements models to define the RUP business model through the business use case and business object models. The strategy proposes a set of activities and heuristics to define conceptual object models starting with models belonging to the client-oriented requirements baseline. The author argues that the use of heuristics in the business model definition encourages pre- through post-traceability between models.

Chapter 7 introduces an approach that is concerned with non-functional features of software systems. The authors examine an extension of UML to capture non-functional information in a way that is similar to its counterpart: the functional information. The NoFun (non-functional) language and the software Quality Standard ISO/IEC9126 are the bases used to achieve some organization about non-functional concepts.

Chapter 8 explores the use of RSL, the language of the RAISE method, to provide formal foundation for UML class diagrams. An automated tool to transform UML class diagrams to RSL is described. Through analysis of the semantics of UML class diagrams and their formal specification in RSL, abstract templates are obtained, which guide the implementation of a translator tool.

Chapter 9 presents the basis of a rigorous process for systematic, object-oriented code generation starting from UML static models. The authors propose an integration of UML, algebraic specifications and Eiffel code. The overall aim of this chapter is to describe the transformation of different kinds of associations to code and the generation of Eiffel assertions.

Chapter 10 describes a step-by-step process for transforming UML class diagrams into entity relation diagrams, which can be used to make objects persistent. The authors also show the possibility of using UML notation to draw entity relation diagrams.

Chapter 11 presents a rigorous approach to specify and check dependency relations between UML models. The authors classify relationships between UML models along three different dimensions: artifact dimension, activity dimension and iteration dimension, and propose a formal description of them. The goal of the proposed formalization is to provide formal foundations for tools that perform intelligent analysis on models used to assist software engineers throughout the development process.

Chapter 12 reviews IS modeling techniques and presents a new technique called “info-mathics” for describing formally hierarchical system architectures. The authors analyze the practical implications of their technique in system analysis and design. They argue that this technique is similar to other engineering techniques applied in well-developed industries.
Chapter 13 proposes the use of Business Process Diagrams (BPD), which are based on UML’s activity diagrams. The authors show how to derive BPD from the business process language of the Event-driven Process Chain (EPC) using Petri nets as a common process metamodel. The authors show examples of business processes and their representation as EPCs and BPDs.

Chapter 14 presents an extension of the UML metamodel with evolutionary stereotypes. An evolutionary stereotype allows the designer to create new classes of UML metamodels with their respective semantics. The authors propose to incorporate evolutionary stereotypes in the tools of modeling so that the developers can modify the UML metamodel. The authors argue that evolutionary stereotypes allow for the automatic generation of code, maintaining the consistence of the UML model.

Chapter 15 describes an extension of the UML activity diagram from workflow. It shows a proposal for the integration of UML with the WfMC (Workflow Management Coalition), Interface 1. The main goal of this approach is to allow UML modeling tools to generate artifacts that represent the workflow process and can be translated to the standard WfMC format. The authors analyze the limitations of UML activity diagrams in modeling an automated organizational process according to the WfMC standard, and then propose how the UML metamodel should be extended to overcome such limitations.

Chapter 16 presents some particularly useful stereotypes to be used in business systems. The authors exemplify their usage with both design drawings and implementation code in C++.

Chapter 17 introduces the CORAS methodology, in which UML and UP are combined to support a model-based, risk assessment on security-critical systems. The authors argue that modeling techniques such as UML contribute to increased understanding by the different stakeholders involved during a risk assessment.

Chapter 18 recommends a set of extensions through a UML profile to support development dealing with safeguarding. The authors take into account three aspects of safeguarding: the business expertise, the interface and the code itself. They show how CORAS can be used for better understanding, documentation and communication during the different phases of the risk management process.

Chapter 19 introduces an extension of RUP with a method that supports the progressive and separate implementation of three different aspects: persistence, distribution and concurrency control. The authors define the software process resulting from the inclusion of this method into RUP, modifying some aspects of the latter. The proposed modifications consider dynamic and static aspects.