Preface

This book presents a special purpose modeling technique for the analysis and design of an important system class, namely form-based enterprise systems. Recent discussions on modeling languages emphasize that there is a strong demand for such domain-specific modeling languages. The class of form-based enterprise systems includes, for example, web shops as well as ERP and B2B solutions and can be said to be paradigmatic for enterprise computing. This book was motivated by the widespread interest in this type of business application from professionals as well as from scientists. The book adapts well-established basic modeling techniques in a novel way in order to achieve a modeling framework optimized for the indicated application domain.

Besides its practical parts the book details theoretical achievements, which lead to real improvements in the application domain of the book. It explains how to model a form-based enterprise system during the analysis and specification phase, and how these models translate into good design. Typical form-based applications have common properties that can be molded into specialized diagram types for such applications. Such a diagram type is the formchart, the central artifact that is described in the book. The formchart is a good example of customized diagrams according to the most recent proposed profiling techniques.

If form-based enterprise systems are modeled with typical general purpose modeling approaches without such customizations, there are a number of obstacles the modeler will face. For example, if one employs use case modeling together with interaction diagrams, the analyst will be confronted with three problems. First of all, the method has to be adapted to the form-based application, since no specific guidance for this special application type is part of the general method. Secondly, the model will become complex for even small-sized problems, since every diagram has to repeat common properties of this very specific application class. Hence the model tends to become highly redundant, and the important distinguishing information is diluted. Furthermore, a third problematic aspect is that current analysis methods are traditionally rather oriented towards event-driven and complex GUI-based applications but not towards form-based applications. Hence the customization demand for form-based applications is particularly high.

Conversely, there are certain benefits the reader of the book can reap by employing the new customized artifacts presented in this book. The reader can obtain faster results and more significant models, because the common properties of enterprise systems are already incorporated in the semantics of the modeling method. The new artifact types presented in this book incorporate the results of studying form-based systems in general, knowledge that a software engineer can hardly obtain in the limited setting of a running project. Our method provides a separation of concerns by splitting the general semantic structure of such applications from the specific information about business logic in the concrete single project. The foundation of the new techniques is fully elaborated in the book for the working developer confronted with everyday problems in professional IT projects. In the same way, the scientist interested in performing novel research on enterprise systems can use this formal reference.

The book is divided into four parts. The first part is a detailed discussion of the new modeling method for form-based systems from a practitioner's viewpoint and explains how the proposed techniques can actually be employed in a project. The second part is about tool support and exemplifies how the concepts introduced in the first part can be exploited by several different implementing technologies. The third part provides the semantic foundation of the different kinds of diagrams and tools introduced in the first and second parts. The fourth part serves as a summary and provides a discussion of related work.

After the introduction the book starts with an in-depth motivation for the new techniques. It is shown that the considered system class encompasses a wide range of important enterprise systems from mainframe/terminal systems through ubiquitous COTS software to modern web applications. The explanations in the book are deliberately based on a realistic running example in order to make a difference. Throughout the book the concepts are exemplified with an online bookshop. This example is not an arbitrary choice of the authors – the important TPC-W benchmark, for example, also uses a standardized online bookshop as a representative example of typical business functionality. The form-oriented information system model is introduced. Different kinds of diagrams for these models, i.e., screen diagrams, page diagrams, form storyboards, and formcharts are introduced for the user interface state part of these models. All of these, and the further model components, i.e., dialogue constraints and the layered data models, are introduced immediately with unambiguous semantics and are used in modeling the running example. Then, techniques for decomposition and refinement are discussed. A parsimonious data modeling language is elaborated. A message approach to the modeling of data interchange is outlined. The book is not primarily about software engineering processes; however, it provides a discussion on how the proposed artifacts can be exploited in an entire software engineering life cycle. The interplay of some proposed best practices that are centered around descriptiveness, artifact orientation, feature orientation, and reuse are discussed. For each concept we show how it can be used to add sustainable value to the respective software engineering activities.

The second part discusses issues of architecture, design, and implementing technology. From this discussion concepts and concrete prototypical technologies for forward engineering, reverse engineering, and the implementation of web presentation layers are derived.

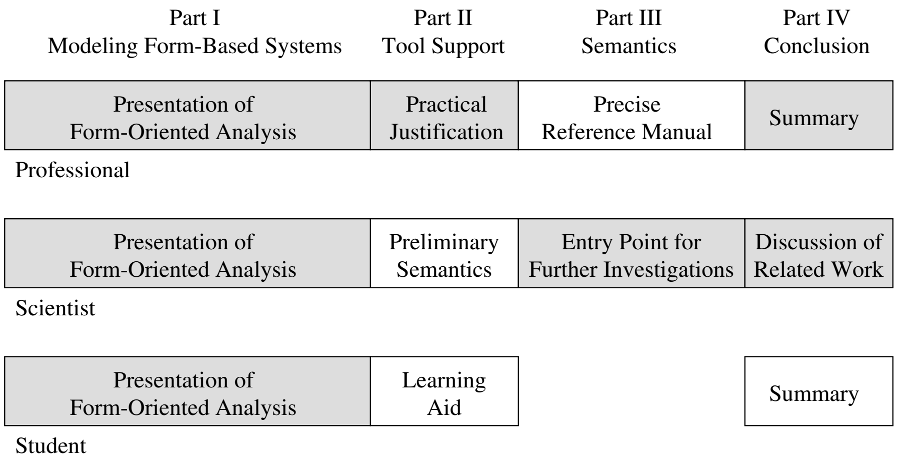
The third part of the book presents the semantic foundation of form-oriented analysis. First, several alternatives for tool support are discussed and given a conceptual basis using an integrated source code model. Then, precise semantics for the form-oriented diagram types are given. For this purpose, a new, lightweight, semantics framework approach is introduced as an alternative to current multi-level metamodeling techniques. Along the lines of the framework approach precise semantics of formcharts, layered data modeling, and the dialogue constraint language are given. This is followed by a discussion of the semantic of the proposed parsimonious data modeling approach. A formal type system for the interplay of server actions and pages of submit/response style systems is provided.

The fourth part provides a focused description of the widely accepted modeling approaches in use. The discussion shows the differences between these approaches and the new method, but it also shows how our method is integrated with standard modeling techniques. For each related method we discuss how it could be applied to enterprise systems and how form-oriented analysis provides a more convenient solution. Therefore this chapter provides a different view on the benefits of form-oriented analysis to the reader. Finally, a summary of the main contributions is provided.

The reader should have some experience with object-oriented programming languages. First-hand experience with visual modeling languages and the graphical tools for them is helpful. Basic knowledge of SQL is also desired for some advanced excursions, but this can be postponed until needed. Related approaches are comprehensively introduced, so that even a reader who is new to these other approaches can follow the arguments.

The book targets professionals, i.e., working software engineers and decision makers, researchers in computer science, and upper-level graduate students who are interested in enterprise systems. Care must be taken, because professionals, researchers, and students typically have different objectives, different dispositions, and different opinions with respect to software engineering topics. This is due to the fact that goals and driving forces are different in industry and academia. Consequently, readers may have different attitudes towards the several parts of the book; see the figure below for a guess. In the figure, supposed main interests are shaded gray, whereas minor interests are left blank.

Professionals actually working on enterprise software will gain a deepened understanding of form-based systems from the abstract system viewpoint pro-



	Part I Modeling Form-Based Systems	Part II Tool Support	Part III Semantics	Part IV Conclusion
Professional	Presentation of Form-Oriented Analysis	Practical Justification	Precise Reference Manual	Summary
Scientist	Presentation of Form-Oriented Analysis	Preliminary Semantics	Entry Point for Further Investigations	Discussion of Related Work
Student	Presentation of Form-Oriented Analysis	Learning Aid		Summary

vided by form-oriented analysis. Many developers already use ad hoc techniques tailored to form-based systems like naive page diagrams or click dummies. These ad hoc techniques arise naturally when developing form-based systems but lack an elaborated conceptual basis. The book allows these developers to strengthen these techniques in practice. Readers can employ the approach directly in projects, because every concept introduced comes with precise semantics and the mutual dependencies between the concepts are elaborated, too. The prototypical forward and reverse engineering tools are suitable for convincing the professional about the potential practical impact of the form-oriented approach. The third part of the book is less important for the professional. If a semantical clarification is needed, this part can serve as a precise reference manual. The professional can use the fourth part as a detailed summary.

Researchers might be especially interested in the third part of the book as an entry point for further investigations. Upper-level graduate students will benefit from the presentation of state-of-the-art knowledge about the development, architecture, and design of enterprise systems in the organizing framework of form-oriented analysis. The second part of the book will help students to grasp more easily the concepts of form-oriented analysis.

Since enterprise applications are a particularly important class of software, almost every IT professional, computer scientist, or computer science student may have some interest in gaining at least an overview of the fundamentals of enterprise computing. The book is written with the different objectives of professionals, researchers, and students in mind. In industry productivity eventually targets return on investment. Product quality and product quantity are limited by productivity. Productivity is limited by the availability of resources. Knowledge acquisition is needed to improve productivity. Academic activity spans two areas that have to be integrated: research and education. While academic research has a subtle target, i.e., the construction of knowledge, higher education has the tangible responsibility to produce well-prepared professionals. Academic research is driven by the pressure to get contributions published in the scientific peer community. Higher education is driven by the demands of the yet uneducated. Altogether these differences result in the following: one and the same concept can be perceived totally differently by individuals in industry and in academia. We encourage all who try to keep an open mind and hope that this book provides valuable information or inspiration.

We are indebted to Martin Große-Rhode for his encouragement and advice. We want to thank our editor Ralf Gerstner for his support and guidance. We would also like to thank the reviewers who made many helpful comments.

Berlin, August 2004 *Dirk Draheim*
Auckland, August 2004 *Gerald Weber*