

Preface

The Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges is about the challenges faced by conceptual-modeling researchers and their successes in meeting these challenges by formalizing underlying theory and showing how to put conceptual modeling into practice. Conceptual modeling is about describing the semantics of software applications at a high level of abstraction. Specifically, conceptual modelers (1) describe structure models in terms of entities, relationships, and constraints; (2) describe behavior or functional models in terms of states, transitions among states, and actions performed in states and transitions; and (3) describe interactions and user interfaces in terms of messages sent and received, information exchanged, and look-and-feel navigation and appearance.

In their typical usage, conceptual-model diagrams are high-level abstractions that enable clients and analysts to understand one another and enable analysts to communicate successfully with application programmers. It is a challenge to successfully provide the right set of modeling constructs at the right level of abstraction to enable this communication. It is an added challenge to formalize these modeling abstractions so that they retain their ease-of-communication property and yet are able to (partially or even fully) generate functioning application software. It is also a challenge to push conceptual modeling toward serving as analysis and development tools for exotic applications such as modeling the computational features of DNA-level life or modeling the ability to read and extract information from free-form text.

A central challenge of conceptual modeling is to facilitate the long-time dream of being able to develop information systems strictly by conceptual modeling. The handbook begins with a manifesto stating that this dream is becoming reality and asserting that applications amenable to conceptual modeling should be programmed abstractly, at the level of conceptual modeling. It thus asserts that “conceptual modeling is programming” and that “the model is the code.” Subsequent chapters support the manifesto’s assertions by showing not only how to abstractly model complex information systems but also how to formalize abstract specifications in ways that let developers complete programming tasks within the conceptual model itself. In addition to addressing this central challenge, several chapters concern demanding challenge areas for conceptual modeling. These include system evolution and migration,

spatial modeling, information integration, conceptual-model-based information extraction, and biological conceptual modeling. The handbook ends with a chapter reflecting on the theoretical foundations of conceptual modeling and addresses both a theory of conceptual modeling (how it is practiced) and a theory of conceptual models (how it is formalized).

Taken together, the chapters selected for inclusion nicely serve the purpose of a handbook by collecting in a single volume many of the best conceptual-modeling ideas and techniques, as well as the challenges that drive research in the field. The handbook is thus suitable as a text for graduate students. It provides a firm foundation for the field of conceptual modeling, and it points toward interesting challenges and puts researchers on a path toward contributing to the conceptual-modeling discipline.

Structurally, the handbook consists of five sections, each with two or more chapters. The first section directly explores the central challenge of conceptual modeling – making conceptual models serve both as high-level abstractions and as executable code. The second section focuses on structure modeling, while the third and fourth sections add material whose main focus is process modeling and user-interface modeling. The final section includes several special challenge-area chapters and ends with central directions for future work both in the theory of conceptual modeling (its practice) and the theory of conceptual models (its formalization).

Section I: *Programming with Conceptual Models*

Chapter 1: Conceptual-Model Programming: A Manifesto. The manifesto expounds upon the vision that all programming activities can and should be carried out completely at the abstract level of conceptual modeling. It asserts that for applications amenable to conceptual-model designs, software developers should never need to write a line of traditional code.

Chapter 2: Model-Driven Software Development. The essence of model-driven software development is the idea that software models can go beyond being mere blueprints: they can constitute the basis for automatically or semiautomatically generating the software system itself. This chapter surveys various major approaches to model-driven software construction and illustrates how model-driven development works in practice.

Section II: *Structure Modeling*

Chapter 3: The Entity-Relationship Model – Toward a Unified View of Data. To provide historical context, this first chapter in the structure-modeling section is a reprint of Peter P. Chen’s 1976 article, which originally appeared as the first article in the first volume of *ACM Transactions on Database Systems*. The publication of this article propelled the ER Model into its place as the foundation for conceptual modeling. No conceptual-modeling handbook would be complete without this historical perspective.

Chapter 4: UML and OCL in Conceptual Modeling. This chapter explains how the Unified Modeling Language (UML) and its accompanying Object Con-

straint Language (OCL) support structure modeling for information systems. UML class diagrams allow information-system-developers to model databases in terms of classes and associations and more advanced features such as part-whole relationships. OCL allows developers to enrich UML diagrams with textual constraints that cannot otherwise be expressed.

Chapter 5: Mapping Conceptual Models to Database Schemas. Automated generation of database schemas from conceptual data models has been a mainstay of conceptual modeling from its earliest beginnings. This chapter generalizes schema-generation rules for use with all kinds of conceptual data models and several types of target databases.

Chapter 6: The Enhanced Entity-Relationship Model. The Enhanced Entity-Relationship Model (EERM) described in this chapter extends the ER Model with complex attributes, cluster or generalization types, and relationship types of higher order. Even with these more complex abstractions, the EERM retains its formal mapping to database schemas and predicate-logic-based integrity constraints. Further, the described EERM extends ER modeling beyond structure modeling to include functionality in terms of queries, transactions, and workflows. The EERM thus facilitates codesign of structure and behavior.

Section III: *Process Modeling*

Chapter 7: Object-Process Methodology for Structure-Behavior Co-Design. Emphasizing both structure modeling and behavior modeling, this chapter asserts that architecting a software system is best approached by codesign using a single model. The chapter describes the Object-Process Methodology (OPM). OPM enables system architects and designers to freely express the tight, inseparable relationships and interactions between a system's static and dynamic components.

Chapter 8: Business-Process Modelling and Workflow Design. Business-process models and workflows provide invaluable understanding of organizational operations. They support process management from modeling and design to execution, monitoring, optimization, and reengineering. This chapter explains basic terms and concepts of process modeling and workflow design and gives details of the three most extensively described process perspectives: the control flow perspective, the organizational perspective, and the data perspective. The chapter also explores some areas of research: problem detection and avoidance in control flow, correctness and generation of process views, and exploitation of temporal information to improve performance.

Chapter 9: BPMN Core Modeling Concepts: Inheritance-Based Execution Semantics. This chapter defines an abstract model for the dynamic semantics of the core process modeling concepts in the Business Process Modeling Notation (BPMN). Each flow element has a rigorous behavior definition, formalized in terms of a basic class inheritance hierarchy that includes sequence flow, flow nodes, gateways, events, and activities. Developers can use the model to test reference implementations and to verify properties of interest for certain classes of BPMN diagrams.

Section IV: *User Interface Modeling*

Chapter 10: Conceptual Modelling of Interaction. Just specifying the structure and behavior of an information system is insufficient – it is also necessary to specify how end users will interact with the system. This chapter presents a practical approach to conceptually specifying end-user interaction. The approach is embedded in a model-driven engineering method, called the OO-Method, which allows full functional systems to be generated from conceptual models. The focus of the chapter, however, is on how the OO-Method supports interaction modeling.

Chapter 11: Conceptual Modelling of Application Stories. The development of complex software systems requires an understanding of how the system is to be used – how actors are to navigate through the system and which actions they are to execute to perform certain tasks. This chapter explains how conceptual models describe navigation paths that correspond to “telling stories” about system usage. The chapter highlights key concepts of storyboarding such as actors, scenarios, and tasks, as well as composed action schemes called “plots.” The chapter also addresses the pragmatics of conceptual storyboards and discusses a development methodology for storyboarding.

Section V: *Special Challenge Areas*

Chapter 12: Evolution and Migration of Information Systems. The management of evolution, migration, refinement, and modernization is an essential component of information-system development. Typical problems include management of evolution and migration; versioning; changes to metadata; system upgrades; modernization in time and space; and change detection, monitoring, and mining. A key challenge is to minimize service disruption and down time and maximize the availability of data and applications. This chapter provides insight into several of these problems from a conceptual-modeling point of view.

Chapter 13: Conceptual Geometric Modelling. This chapter presents a geometrically enhanced ER Model (GERM), which preserves the key principles of ER modeling while at the same time introducing bulk constructions and types that support geometric objects. GERM distinguishes between a syntactic level of types and an explicit internal level, in which types give rise to polyhedra defined by algebraic varieties. GERM further emphasizes the stability of algebraic operations by means of a natural modeling algebra that extends the usual Boolean operations on point sets.

Chapter 14: Data Integration. Data integration is about combining data residing in different sources (virtually or actually) and providing users with a unified view of the data. At the heart of data integration is conceptual matching and mappings, making conceptual modeling of one sort or another central to data integration. This chapter presents an overview of the relevant research activities and ideas in data integration, and it discusses as an example the MOMIS system – a framework to perform information extraction and integration from both structured and semistructured data sources.

Chapter 15: Conceptual Modeling Foundations for a Web of Knowledge. The first-generation web is a web of pages. This chapter shows how conceptual modeling

can play a central role in turning the web of pages into a web of knowledge to be superimposed over the current web of pages. Success in this endeavor would enable the web to become a gigantic, queriable database.

Chapter 16: A Conceptual Modeling Approach to Improving Human Genome Understanding. The main purpose of conceptual modeling is to represent knowledge in some application domain – usually for the purpose of developing and maintaining information systems. In a slight paradigm shift, it is possible to imagine the human body as an information system – highly complex, but ultimately built on biological information-processing molecules. This paradigm shift allows for exciting possibilities: just like acquiring the source code of a manmade system allows for postproduction modifications and easy software maintenance, the same could very well apply to the human body. Acquiring the source code to the human information system begins with the first step in any information system development – creation of a comprehensive and correct conceptual model of the human genome. This chapter aims at this objective.

Chapter 17: The Theory of Conceptual Models, the Theory of Conceptual Modelling, and the Foundations of Conceptual Modelling. This final chapter aims at the heart of conceptual modeling itself. It not only summarizes the foundations of conceptual modeling, but it also goes beyond this toward the development of both a theory of conceptual models and a theory of the practice of conceptual modeling. A remaining challenge for conceptual modeling is to harness its foundations in terms of a theoretical perspective that leads to better usage by practitioners and pushes researchers to meet its challenges head-on and resolve them.

Provo, Utah, USA
Kiel, Germany
August 2010

*David W. Embley
Bernhard Thalheim*

Handbook of Conceptual Modeling
Theory, Practice, and Research Challenges
Embley, D.W.; Thalheim, B. (Eds.)
2011, XIX, 589 p., Hardcover
ISBN: 978-3-642-15864-3