Data Modeling Education: The Changing Technology

Michael A. Chilton
Roger McHaney
Bongsug Chae
Department of Management
Kansas State University
Manhattan, Kansas 66506
mchilton@ksu.edu , mchaney@ksu.edu , bchae@ksu.edu

ABSTRACT

Data modeling is a difficult topic for students to learn. Worse yet is the fact that practitioners, who look to academia for methods and techniques to perform such model building have found little on which to standardize, although many techniques exist. Entity relationship (ER) modeling was developed in order to help database developers visualize their (relational) database design with its data stores and internal relationships. This technique was certainly an important step forward, yet data collected over the past 11 years would indicate database developers are still having difficulty learning, assimilating, and using design techniques (cf. Blaha, 2004). Confounding the issue is the arrival of the object-oriented paradigm. The Unified Modeling Language (UML) was introduced in order to speed, simplify, and clarify design of systems. Portions of the UML are derived from ER modeling and are useful in merging the front end portion of the system with the back end data storage so a picture of the entire system can be viewed by the designer. While providing functionality that ER modeling lacks, the UML approach to data modeling also leaves some developers indecisive and confused as to which technique to use in practice. The same indecision appears to haunt the academic world. So how should data modeling be taught? In order to shed light on this question, we asked contributors to focus on whether this new system of modeling (the UML) yields a better understanding of the database design to the extent that better database designs result. We detected a buzz in the literature and in the IT world that a dichotomy of opinion over this question exists, and so this special issue was born. Educators need to air their opinions, facts, and results and discuss this controversial topic to encourage refinement in this important area. We hope that research ideas can be generated and practitioners informed that this topic is being addressed in academia. As expected, the contributors to this issue provided a dichotomy of opinion but surprisingly, their experiences and opinions moved the issue in a direction far different than what we could have predicted. We now provide you with insight into this poignant topic by presenting this special issue.

Keywords: Data modeling, UML, ER models

1. INTRODUCTION

Chen (1976) developed the entity relationship diagram (ERD) as a set of tools to assist the database designer in visualizing the internal workings of a complex design. Chen laid out the basic data model with symbols and later other extensions were added. The extended ER model was developed by Teorey, Yang and Fry (1986) to include the concept of generalization (inheritance) in the modeling technique. No clear standard has emerged and there are several in use, including the so-called Crow's foot model, or Information Engineering and the Integrated Definition for Information Modeling (IDEF1X) adopted by the Federal Information Processing Standards agency in 1993.

Object-oriented (OO) techniques began to infiltrate the database world because it seemed that this natural way of processing software "objects" in OO programming languages could be extended to the database. In fact, it seemed like a perfect match since each record in a database table fits the definition of an object as defined in the OO paradigm. Database designers rushed to market their new product, the OODBMS, because at the time, the writing was (or at least seemed to be) clearly on the wall—all software will eventually be subsumed under the OO umbrella. Growth rates of these products were predicted to be so high (~50%) that one wonders how manufacturers could have kept up. The harsh reality was this growth rate was never achieved, and the OODBMS approach has never completely caught on. Pockets of use may be in existence, but only because the technology has been suited to specialized applications—CAD/CAM, multimedia systems and network management systems as examples. The OODBMS has not effectively competed with the advantages provided by relational database's particularly strong roots in mathematical theory. Added to this is the extent of infrastructure held by the relational products making it is extremely doubtful it will be supplanted by the (pure) OODBMS in the foreseeable future.

In spite of many misgivings, the OO movement did force relational database designers to add object-oriented extensions to their products. Primarily due to the need to process complex objects, the object-relational database is now able to store and perform searches within audio, geographic data, telecommunications data[1] and other complex data types. The OO movement also brought new data modeling tools. The Unified Modeling Language (UML) became the standard for OO systems and included tools for database design. The UML class diagram is probably the strongest contender in this realm since it maps directly into a relational (logical) design and is able to convey even more system information than just entities and their relationships. The use of UML diagrams by relational database designers is somewhat controversial and not entirely accepted. We detected this controversy among database instructors and systems analysis instructors and decided looking further into this issue would help to place the arguments for both sides on the table and foster a healthy academic discussion. Our goal for this issue was to advance the body of knowledge on the use of modeling techniques by airing this controversy and promoting cogent discussion of the topic. We hope this goal has been accomplished.

2 DATA MODELING

Since the advent of E.F. Codd's relational data model in 1970, database users have sought to represent their perceptions of stored data to both make sense logically in terms of the real-world person, event, place, or thing about which information is required, and mathematically to ensure valid and accurate storage of data. These perceptions have been captured, independent of hardware and software constraints using data models to conceptually represent data structures that include the data, and associations between these data. Therefore, data models are concerned with what data is required and how that data should be organized as the gap between reality and a database representation is bridged. It is helpful to think of a data model as a blueprint for the construction of a database.

2.1 ER Approach to Data Modeling

Chen's data modeling technique attempted to unify different views of data obtained with the network database view and the newer relational database view and to address the "semantic ambiguities" (Chen, 1976, p. 9) imposed by each. The new ER model used a visual representation of the data objects called an Entity-Relationship Diagram (ERD) to provide a conceptual model of real world entities and relationships.

The ER model offered several advantages over other competitor modeling techniques. These advantages are particularly relevant with working with relational database systems and include:

- Ease of understanding: The ER model can be understood by both database designers and users needing support provided by the database system. Minimal training is required, thus making the ER model ideal for the facilitation of communication between users and designers (cf., Watson, this issue).
- Ease of transformation: The ER model very closely approximates the construction of relational database tables and is consistent with its mathematical underpinnings. This allows for easy implementation into a specific DBMS.[2]

The primary objects in ER modeling are entities and relationships. Entities are generated by starting with those recognizable objects about which data is collected and stored. These more obvious entities include people, places, things, and transactions. Less obvious entities can be found as the strong entities (those that do not depend on other entities for existence) are related to one another. The process of relating entities may uncover the need for additional (weak) entities whose existence results from the existence of other entities and/or can be used to map a specific relationship.

Entities are described by characteristics, called attributes that provide values for specific instances within each entity class. For instance, a CUSTOMER entity can be described by attributes such as Name (Mary Parker), CustomerID (2002002), and Address (123 Happy Lane). Care must be taken not to include attributes within the entity that actually describes a relationship with another entity or that should be a part of another entity altogether. The inability to recognize which facts become characteristics of which entity creates confusion in database designers and therefore increases the difficulty of creating a model that mirrors the users' perceptions of reality. The irony of the situation is that the ER modeling technique is supposed to meliorate this problem, yet our experience is that this is one of the most difficult portions for the beginning data modeler to assimilate. Database designers and instructors are continually looking for better methods and techniques to help improve understanding of the logical model. Several papers in this issue are dedicated to this proposition.

2.2 UML Approach to Data Modeling

The roots of UML can be traced to 1994 at Rational Software Corporation where Grady Booch and James Rumbaugh (formerly of General Electric) reconciled features of the Booch method and Object Modeling Technique (OMT) to begin work on a unified modeling language. In 1995 Rational Software Corporation aquired Objectory and with it Ivar Jacobson, the developer of the object oriented software method (OOSE). The resulting synergy provided foundational work for UML which was further developed and formalized by a consortium of leading software companies and practitioners.

The UML is a comprehensive set of standards specifically developed to support software engineering aspects of large

[1] Consider the image, audio and video extenders now available for IBM's DB2 product.

[2] It also might be considered a detriment since it obscures the differences between the logical and physical models (cf. Kroenke, this issue).

systems using object oriented programming languages. It can be used both as a development tool and for communicating with end users within the development environment.

The UML language should not be confused with the system diagrams it aids in generating. Each diagram contributes in part to a graphic depiction of a system's model. The system's model comprises three important aspects, the functional model, the object model, and the dynamic model. The functional model relates to the end user's perspective and relies on the Use Case Diagram. The object model provides system structure using objects, attributes, and associations. The Class Diagram is most commonly here. The dynamic model provides details regarding the system's internal workings and is developed with Sequence Diagrams, State Machine Diagrams, and Activity Diagrams.

The class diagram is commonly used for data modeling with the UML. As the name implies, the class diagram depicts classes within a data model. When viewing a system from an OO perspective, classes have member variables (or attributes), member functions (operations), and associations (relationships) with other classes. This roughly translates into an ER model, and is therefore often purported to be sufficient for data modeling.

3. ISSUE OVERVIEW

We present ten papers in this issue. In the first paper, we hear from Jim Rumbaugh, one of the three co-inventors of the UML. He succinctly points out that UML was developed with entity relationship modeling in mind, thus it is an extension of Chen's original ER Diagram. He discusses, compares and contrasts ER modeling with object-oriented UML modeling and hits on some topics that both database instructors and systems analysis instructors should keep in mind as they teach these complex and important subjects, not the least of which is that there seems to be a (needless) dichotomy between systems analysis design and database design.

We have four papers from database textbook authors: Mannino, Kroenke (co-authored by Jim Gray), Watson, and Connolly and Begg. Mannino argues that both OO and relational techniques should be taught in IS curricula, albeit separately. Kroenke and Gray argue that neither technique precisely meets the need. While Mannino bases his arguments on the student, Kroenke and Gray take a more database-oriented approach. Watson and Connolly and Begg present viewpoints that the technique one uses is not as important as the assimilation of the fine art of data modeling. Each paper provides useful information based on their experiences in the classroom.

In addition to the textbook authors, several teams of academic researchers and industry practitioners weigh in with their experiences and opinions regarding teaching. In their contribution, Traci Carte of the University of Oklahoma, Jon Jasperson of Texas A & M University, and Mark Cornelius of Anadarko Petroleum Corporation describe a teaching approach that draws from both database and systems analysis classes. They conclude that "existing knowledge of structured systems development can and should inform [their] teaching processes when teaching object-oriented systems development techniques." They support this with an example from industry that demonstrates the need to "easily shift from structured to object-oriented thinking."

Another set of authors, Hsui-lin Winkler from Pace University and Henning Seip from SkillPROOF, Inc. move even further from the academic aspects of UML versus ERD and investigate employment demands in data modeling. They report that data modeling remains a required skill in an analysis of employment data published by individual companies collected from 137 IT-focused companies since the beginning of 2004. Most employers do not list specific required data modeling methods in job postings but often list them as desired skills. When a job posting includes a requirement of competence in UML, it is typically found in conjunction with software engineering and application development. Conversely, competence in using an ERD is most commonly mentioned in conjunction with database design and maintenance.

Ming Wang argues in paper #5 that teaching the UML within a database course is easily learned so long as the students have been prepared by first completing a course in object-oriented programming and a course in database design. The foundation of her teaching is based upon the temporal progression of the methods, i.e., the ER technique was produced first and the techniques rooted in the UML were produced later. Her experience provides support for this temporal progression—students should learn ER techniques first before they attempt to assimilate techniques that provide different views and more information. She finds that this works well in the classroom.

Donald Golden and Victor Matos present the IS program at Cleveland State University that has adopted the UML as a tool for data modeling. They posit that as IS development is becoming larger and more complex than before, the system development should be viewed as an integrated process spanning from understanding of user and business requirements to the development of logical and physical models and the implementation of the physical system. From this, they argue that the ERD has become less useful because it is limited to database design, but that techniques based on UML have become more effective for linking systems analysis and database implementation and thus helps students acquire a more accurate view of IS development as the integrated process.

James Suleiman and Monica Garfield offer a picture about the current status of conceptual data modeling techniques in database education. They examine the level of support for UML vs. ER notations for a tool for data modeling in undergraduate database courses in the U.S. They gather the data for their study through (1) an analysis of textbook and data modeling software tools and (2) a survey of data modeling techniques used in the undergraduate IS programs at nineteen major universities in the U.S. Their study found that the use of UML is not as popular as many practitioners and academicians might expect and that the support of UML among database course teachers does not appear to be very strong. In the light of this finding, the authors discuss the strengths and shortcomings of UML for teaching conceptual data modeling.

4. LESSONS LEARNED

While the publication of this issue and the arguments contained within each paper will not decisively end the debate regarding the teaching of data modeling techniques, it does provide rich and varied insights into the current state-of-the-art. Practical considerations make the total switchover to the UML premature, yet the need for improvement and forward thinking preclude using ER modeling techniques indefinitely, especially considering the admonition of Kroeke and Gray that both of these techniques fall short with respect to certain aspects of data modeling. These aspects are punctuated not simply with newer technology that allows the storage and manipulation of complex data types, but with the need to resolve heretofore intractable problems (e.g., multi-valued attributes) in better ways.

What we have discovered in our goal of airing differences between two techniques is that the real controversy is a parochial one—that there is a discord between systems analysts and database designers. In one camp we have the SAs who are comfortable with object-oriented techniques, programming languages and methods, while in the other camp we find data modelers and database designers who believe that the technique should be rooted in mathematical theory and the ERD is the best tool to use. In addition, the database camp argues that the adding of (business) processes (data flows) to the modeling tool only tends to skew the design to fit these processes—a fundamental violation of the concept of data independence. We believe that systems analysis and database design are not mutually exclusive but should be part of a holistic IT paradigm. We hope that most educators and practitioners would agree that a holistic teaching focus would come in a step-wise fashion analogous to teaching young students basic skills in arithmetic. The student must master the addition, subtraction, multiplication and division tables before moving on to the calculator. By the same token, IT students must master the components taken individually to form a basis that can grow as new knowledge is gained. Where we believe that additional research and discussion is needed is in the sequence and the content of these components.

5. REFERENCES

Blaha, M. (2004). "A Copper Bullet for Software Quality Improvement," *IEEE Computer*, (37:2), pp. 21-25.

Chen, P. P. (1976). "The Entity-Relationship Model—Toward a Unified View of Data," *ACM Transactions on Database Systems*, (1:1), pp. 9-36.

Teorey, T. J., Yang, D, and Fry, J. P. (1986). "A logical design methodology for relational databases using the extended Entity-Relationship model," *Computing Surveys* (18:2), pp. 197-222.

AUTHOR BIOGRAPHIES

Michael Chilton is an Assistant Professor in Management Information Systems at Kansas State University, where he teaches database design and telecommunications and networking. His teaching and research interests are in database modeling and design, management of IT personnel and computer communications and networking. He has published in the *Database for Advances in Information Systems*, the *Journal of Management Information Systems* and other journals. Prior to entering academe, he worked for the US government specializing in homeland defense.

Roger McHaney is a Professor of Management Information Systems at Kansas State University. He received his Ph.D. in Computer Information Systems and Quantitative Analysis at the University of Arkansas in 1995. His research interests include applications of end-user satisfaction, computer simulation, the simulation lifecycle, database applications, and effective use of computers in rural settings. He has authored a computer simulation textbook and numerous technical articles in publications such as *Decision Sciences, Information & Management, Decision Support Systems,* and the *International Journal of Production Research*.

Bongsug Chae is an Assistant Professor of Management Information Systems at Kansas State University, where he teaches systems analysis & design, computer programming, and management science. He has published in *European Journal of Information Systems, Journal of Strategic Information System*, and other journals. His research interests are enterprise systems design and implementation, business process & knowledge management, technology innovation, and supply chain management.

Information Systems & Computing
Academic Professionals

Serving Information Systems Educators

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096