Journal of Information Systems Education, Vol. 17(1)

Notation Usage in Data Modeling Education

Michael Mannino
The Business School, P.O. Box 173364
University of Colorado at Denver
Denver, CO 80217-3364
Michael.Mannino@cudenver.edu

ABSTRACT

In the absence of a well-established data modeling standard, the best practice in information systems education is to teach both ERD modeling and UML data modeling. The justification for this position is the lack of inherent advantages of either approach, the importance of training students for existing practice, and the low prospects for future establishment of a unifying standard. To minimize student confusion, ERD data modeling and UML data modeling should be taught in separate courses. In teaching both approaches, the goals should be precise usage of notation, exploration of alternative designs, and recognition of design errors as well as the difficulty of capturing data requirements in unstructured business situations.

Keywords: Entity relationship diagram (ERD), Unified Modeling Language (UML), CASE tools, Design errors, Design transformations

1. STATUS OF DATA MODELING NOTATIONS

In the history of graphical data modeling notations, no standard notation has emerged despite large amounts of intellectual effort and commercial development. The IDEF1X standard (FIPS, 1993) developed by the U.S. federal government as an information processing standard in the early 1990s, is the most significant attempt at standardization. However, the IDEF1X standard has received only limited acceptance primarily in government contract work.

Diversity of notation remains the rule in data modeling as evidenced by leading database textbooks and commercial CASE tools. Leading database textbooks use different ERD notation typically a notation based on the original Chen notation, the information engineering notation, or the IDEF1X notation. Some database textbooks also present the UML data modeling notation in a separate chapter on object-oriented modeling. Major CASE vendors typically emphasize one ERD notation but support a variety of ERD notations. IDEF1X seems to have the most support in CASE tools but the level of support does not come close to a standard. Most major CASE tools also support the UML but the level support for the UML for database development varies widely.

The diversity of data modeling notation in textbooks and CASE tools is matched by widely varying practice in commercial data modeling. Although I have not seen credible surveys about data modeling usage, I surmise that practice is at least as diverse as CASE tools and database textbooks. With large organizations, CASE tools may have a large influence on the data modeling notation used. Government contracting also can influence the data modeling notation especially for IDEF1X and UML in some cases. For medium and small organizations, CASE tool usage may not be prevalent so the data modeling support by a relational DBMS vendor may be used in place of a CASE tool.

Given the diversity of data modeling notation in usage now, the prospects for a standard seem remote at least in the near future. UML is the standard for large-scale software development, but it is not a standard for business systems. No CASE product has the dominance to establish a data modeling standard. No standards organization has initiated an effort with wide support to establish a data modeling standard.

The advantages of the UML notation (inheritance support and integrated notation) are not compelling enough in business systems development to make the UML a data modeling standard. Inheritance in business data modeling is primarily used for data types. User-defined data types were well-established in most major DBMS products before the SQL:1999 standard. The usage of generalization for tables seems specialized as few vendors support the table generalization features of SQL:2003. The integration of data modeling, function modeling, and event modeling are compelling features of the UML. The integration is important for business systems that involve large amounts of object-oriented programming language code. For other business systems with demanding form and report requirements, the integration of the UML is not compelling.

2. IMPLICATIONS FOR DATA MODELING EDUCATION

Given the lack of a standard data modeling notation, students should be prepared for diversity of notation during their career. Students should be exposed to an ERD notation and the UML notation, preferably in separate courses. Typically, ERDs are taught in a database course and UMLs in a systems analysis and design course. Some instruction about ERDs should expose students to alternative ERD notations. To provide more depth, students can use the notation of choice in a capstone course or degree project.

Beyond exposing students to alternative data modeling notations, instruction should emphasize precise usage of notation, identification of design errors, and generation of alternative designs. Precise usage of notation is sometimes ignored because CASE tools enforce a level of structural integrity. However, students often need help with subtle details of structural integrity involving identification dependency, generalization hierarchies, and foreign keys. Students should be encouraged to use features of a CASE tool to check for violations of structural integrity. For example, the ER Assistant available from Irwin McGraw-Hill provides a check diagram feature to identify structural errors in an ERD.

Design errors involve inconsistencies between a problem narrative and a data model. The biggest impediment to instruction about design errors is an attitude that data modeling is entirely subjective. Students often substitute their judgment and experience for the details in a specification. Students need clear instruction to demonstrate inconsistencies between a specification and a data model. After students understand the goal of consistency with a specification, design errors should be easier to identify. Students can then learn about using design documentation to record incompleteness and ambiguity in a specification. CASE tools do not provide assistance with design errors so enough problems must be discussed to provide a sufficient knowledge base.

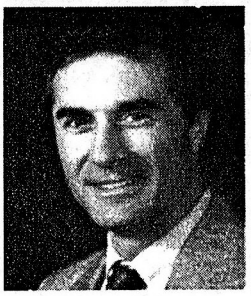
Design transformations provide a tool to generate alternative designs. Students should be told about the importance of design simplicity especially for an initial data model. Design transformations provide a structured way to support alternative data models. Students should be instructed about the details of each transformation along with the reasons to use each transformation. CASE tools do not seem to support design transformations so enough problems should be discussed to enable students to apply the transformations.

These guidelines for data modeling education (exposure to alternative notations, precise usage of notation, identification of design errors, and design transformations) are just building blocks to develop data modeling skills. These guidelines must be supplemented by analysis of complex business situations in projects and case studies. Ideally, projects should be given in introductory courses as well as larger projects in capstone courses or degree projects. If sufficient time is not available in courses for projects, case studies derived from real business situations can be used. To prepare students to work on complex case studies and projects, class discussion should involve difficulties of obtaining a database specification. Students should understand the need to interact with multiple stakeholders, narrow the scope of a proposed database, ignore irrelevant details, resolve conflicts, and complete missing requirements as a project progresses. Peer review and instructor mentoring can help students apply data modeling guidelines as well as grapple with the difficulties encountered in complex business situations.

3. REFERENCES

FIPS (1993), *Federal Information Processing Standard 184*, "Integration Definition for Information Modeling (IDEF1X)," National Institute of Standards and Technology, December 1993, available for download from http://www.4dcompanion.com/downloads/standards/IDEF1X.pdf.

AUTHOR BIOGRAPHY



Michael V. Mannino is on the Information Systems faculty of the Business School, the University of Colorado at Denver. Previously, he was a faculty member in the Computer and Information Sciences Department at the University of Florida, the Department of Management Science and Information Systems at the University of Texas at Austin, and the Management Science Department at the University of Washington. Dr. Mannino teaches and conducts research in database management, software development, and data mining. His articles have appeared in journals affiliated with the ACM, IEEE, and TIMS. He is the author of the textbook, *Database Design, Application Development, and Administration*, published by Irwin McGraw-Hill.

Information Systems & Computing
Academic Professionals

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