The Impact of Conceptual Data Models on End-User Performance

Prashant C. Palvia
Chechen Liao
Pui-Lai To
Memphis State University

While implementation/logical data models have been extensively studied and reported on, there is relatively less attention on the conceptual data models, especially from an end-user empirical perspective. Conceptual models are more suited for end-users due to the richness in semantic expressiveness and user-oriented features, but usually are not directly implemented. In this article, we examine three conceptual models: data structure diagram, entity-relationship model, and object-oriented model from the viewpoint of end-users. Results of two empirical studies, one experimental and one survey, are described. A comparative examination of the three data models on comprehension, efficiency, productivity, and a whole host of other characteristics has been made. The general evidence from the experimental study is that the user performance is much superior in terms of comprehension, efficiency, and productivity using the object-oriented model than the data structure diagram or the entity-relationship model. The second study suggests that this clear user preference for the OOM model diminishes with increased computer and database experience. Given the explosive growth in recent years of end-user computing and their use of databases, the findings of this study should be of great concern for users as well as information systems specialists.

End user computing (EUC) has undergone explosive growth in the past decade (Benson, 1983; Cheney et al., 1986; Henderson and Treacy, 1986; McLean, 1977; Panko, 1987). It has been aptly described as “a rapidly growing and irreversible phenomenon” (Alavi and Weiss, 1986). End users are involved in two distinct computing activities: development vs. use (Sipior & Sanders, 1989). In both modes, a primary requirement is the ability to extract and manipulate data that is stored in organizational databases. This task performance is facilitated and greatly influenced by the underlying data model. There are two widely known classes of data models: logical/implementation models, and conceptual models. This article reports on two related empirical studies: the first experimental study investigates user performance with three conceptual models, and the second survey study describes user perceptions of the three models. The two studies, taken together, have important implications for the use of conceptual models by end users.

In the next section, we describe several studies in end user computing that are relevant to our article. Section three describes data models in general, three conceptual data models in particular, and reasons for the focus on conceptual data models in this article. Section four provides an overview of related research. The next two sections describe the two studies. Included in each study are the study objectives, methodology, analysis, and results. Finally, implications of the two studies and future research directions are described in the last section.

End User Computing/Novice Users

Many researchers have addressed the role of end users as developers of information systems and database applications (Gremillion and Pyburn, 1983; McLean, 1979). Database technology is readily available and being often used by end users for application to nontrivial problems (Batra et al., 1990). When Rockart and Flannery (1983) in their seminal article categorized end users into six distinct categories, three important categories were: non-programming end users, command level users and end-user programmers. These three types include a vast number of end users, and they all interface with data stored in databases. In the non-programming mode,
they simply access and view data. In the command level mode, they need to access data on their own terms, and perform simple inquiries often with simple calculations. In the programming mode, they interface with databases while developing programs in a high-level language, or a fourth generation language (4GL). Other categorizations of end users are: development vs. use (Sipior & Sanders, 1989), and individual vs. group (Sipior & Sanders, 1989) or personal vs. organizational (Amoroso, 1988). Again in any of these classes, the need for data retrieval and analysis is clearly recognized. In terms of the relative importance of data to end users, an often cited study on end user satisfaction (Doll & Torkzadeh, 1988) reported five factors, three of which are data related (i.e., content, accuracy, and format).

Despite the incessant growth in end user computing (EUC), research results have identified several specific pitfalls and risks associated with end-user computing and end user developed applications (Amoroso, 1988; Alavi & Weiss, 1986; Davis, 1982). Alavi and Weiss (1986) classified such risks and suggested the development of controls to counter the risks. Chrisman and Beczyce (1987) blamed many of these application problems on poor database design, and suggested effective communication with the users as a solution. O’Donnell and March (1987) classified EUC problems and their impacts on productivity into four categories. The recommended solutions to the problems frequently include well-designed controls (Alavi & Weiss, 1986; Amoroso, 1988; O’Donnell & March, 1987). These problems and controls include areas directly relevant to this article, i.e., information architecture, software design process, conceptual design, data administration, and data integrity.

Data Models

The data is structured and presented to its user in the form of a data model. The general database literature, published books (the most accessible source of information for most users), and commercial software products have largely emphasized the three logical/implementation models: the hierarchical model and the network model (Kroenke, 1992; McFadden & Hoffer, 1988), and the relational model (Date, 1990). Of these three, the relational approach represents the dominant trend in the marketplace today (Date, 1990). All three are record-based models, have semantic limitations as explicated by Kent (1979), and include several implementation features. It is very difficult, sometimes impossible, to capture certain semantics of the real world into these models. What is required from the user’s viewpoint is a data model which is powerful in its semantic expressiveness, provides a user-friendly interface, provides a good communication medium for user data requirements, and is compatible with the user’s view of the world. Conceptual data models attempt to meet these characteristics by including more semantic information and eliminating/de-emphasizing the physical constructs.

Primary examples of conceptual data models are: entity-relationship model or ERM (Chen, 1977), data structure diagram or DSD (Bachman, 1969), semantic data model or SDM (Hammer and McLeod, 1978), and object-oriented model or OOM (Kroenke & Dolan, 1988; Kroenke, 1992; Heintz, 1991; Kim, 1990a; Kim, 1990b; Bertino & Martino, 1991). In this article, we evaluate DSD, ERM, and a version of OOM from a novice user perspective. SDM is not included as many of the SDM concepts have been incorporated in OOM. In the authors’ experience and in the literature, DSD and ERM seem to be most widely used, and OOM is beginning to receive attention. Fairly standardized descriptions of DSD and ERM are available in the literature, as per our earlier references. The object-oriented model is new and still evolving; as such it is neither standardized nor fully defined at this point. Note, however, that most descriptions of the object-oriented model have underscored modeling richness and more natural representation of entities in the user’s world (Kim, 1990b; Heintz, 1991; Kroenke, 1992).

Given the absence of a standard OOM model, we have used the object-oriented approach as described by Kroenke and Dolan (1988) and subsequently by Kroenke (1992). Kroenke’s database book is widely adopted in many schools for their databases classes; as such his OOM version has received wide exposure. The approach was first described by Kroenke and Dolan (1988) where they provided a definition of a database object, described object characteristics, and discussed five types of objects. In the latest revision of the book, Kroenke (1992) calls the objects as semantic objects and the model as semantic object model (to distinguish from object-oriented programming). An abbreviated description of Kroenke’s version of OOM and objects follows; for a complete description, see Kroenke and Dolan (1988) and Kroenke (1992).

An object is a named collection of properties that sufficiently describes an entity in the user’s work environment (Kroenke & Dolan, 1988). An entity is something a user perceives as an independent unit, meaning it can stand on its own. An object is a structure that represents an entity; an object instance is a particular object. An object diagram is used to summarize the structure of objects in a visual portrait-oriented rectangle form. Figure 1a and 1b are diagrams for a department object and a department instance object, as reproduced from Kroenke (1992). Depending on the type of properties and composition of objects within an object, six types of objects are defined: simple objects, compound objects, association objects, hybrid objects, and generalization and subtype objects. Thus, this OOM version does include many features of OOM models, but not all (Kim, 1990b). Particularly missing is the encapsulation of methods with the data; however, the focus of our study is the evaluation of the