OPOSSUM: A FLEXIBLE SCHEMA VISUALIZATION AND EDITING TOOL

Eben M. Haber, Yannis E. Ioannidis, and Miron Livny

Department of Computer Sciences
University of Wisconsin, Madison
1210 Dayton Street, Madison, WI 53706
E-Mail: {haber,yannis,miron}@cs.wisc.edu

ABSTRACT

In the spirit of interdependence of the different areas of CHI research, we present a description of OPOSSUM, a visualization tool inspired by concepts from heterogeneous databases. OPOSSUM is a tool for visualizing and editing structured data; we use it to view and modify object-oriented database schemas. OPOSSUM is based on a formalism that allows declarative descriptions of the following: 1) a model describing the schema to be visualized, 2) a model describing visualizations, and 3) a mapping between the two models. The formal approach makes OPOSSUM very flexible, and promises solutions to several problems of schema visualization.

KEYWORDS

Model-based Interface Tools, Metaphors, Database, Schema Visualization

INTRODUCTION

This paper describes OPOSSUM (Obtaining Presentations Of Semantic Schemas Using Metaphors), a structured data visualization and editing tool currently under development. OPOSSUM's role is visualization of object-oriented database schemas, although it works with other structured information. OPOSSUM is based on a formalism that describes visualization; the formalism was inspired by the concept of schema mapping from the field of heterogeneous databases [3]. This formalism defines visualizations through three declarative descriptions: a *data model* that describes the schemas to be visualized, a *visual model* that describes visualizations, and a *metaphor* mapping between the two models. OPOSSUM now in prototype form; it allows a schema to be edited through manipulation of the schema's visualization.

OVERVIEW OF OUR FORMALISM

OPOSSUM is based on a formalism that describes transformations of structured information between abstract and visual forms. This formalism is fully described in [1]. We developed the formalism to improve visualization flexibility; there are several different useful visualizations for any given schema, thus flexibility is required.

Database schemas are instances of the data model used by a database system. For example, each schema in a relational database is an instance of the relational data model. In general, a schema is an instance of some information model. The process of visualizing a data model schema (i.e., converting an instance of a data model to a visual form) is similar to schema mapping in a heterogeneous database (i.e., transforming an instance of one data model to an instance of another data model) [3]. Our formalism grew from this idea; instead of mapping schemas between different data models, it maps data model schemas to schemas of a visual model. A visual model is distinguished in that visual model schemas define an appearance for themselves. For example, one might want to map a relational schema to a visual model schema formed of boxes, text, and lines.

Information models are defined declaratively in terms of *types of primitives* in the model, *attributes* of the primitives of those types, allowed *values* of the attributes, and *constraints* that must be satisfied by schemas of the model. For example, the relational data model has two types of primitives: *relation* and *attribute*, the former having the attribute *name*, and the latter having the attributes *name*, *type*, and a *relation* with which it is associated. Metaphors are defined as the union of three functions that map from: 1) visual model types of primitives to data model types of primitives, 2) visual model attributes to data model attributes, and 3) visual model values to data model values. For example, one type of visual model primitive could be defined to be a box with a piece of text inside. A metaphor could specify that this type of primitive corresponds with *relation* in the above data model, and that the text field corresponds with the *relation name*. Given a metaphor between a data and a visual model, it is possible to induce a mapping between schemas of the two models. Metaphors ensure a structural correspondence between schemas of the two models such that, when users view a visual model schema, they can (with knowledge of the metaphor) infer the corresponding data model schema. The metaphor defines the meaning of visual model features with respect to data model features. Metaphor mappings may be many-to-one, indicating that several visual model features have the same meaning with respect to the data model; this provides representation choice (two types of primitives or attriute

values that may be used interchangeably) or redundancy (two attributes representing the same information).

Metaphors may be tested for correctness. In this context, correctness means that any data model schema can be unambiguously mapped to a visual model schema and back again. In addition, the formalism allows metaphors to be composed, forming new, richer metaphors with increased choice or redundancy.

OPOSSUM, A FORMALISM BASED TOOL

OPOSSUM is designed as the core of a graphical user interface for an object-oriented scientific database system. Its first role is to allow schema display and editing via direct manipulation of a schema visualization, though it may be extended in the future to support visual querying and data display. It is implemented in C++ and InterViews/Unidraw, running with X-windows on a Unix workstation. Currently, OPOSSUM permits schema definition and editing for any models and metaphors supported by our formalism.

OPOSSUM is very flexible in order to work with arbitrary models and metaphors. Descriptions of models and metaphors are maintained as data structures in main memory, though they are saved to disk between sessions. The OPOSSUM interface consists of one or more windows, each associated with a particular visual model (and usually a corresponding data model and metaphor). Each window includes tools for creating primitives of the associated visual model. In addition, there is a general attribute editing tool that pops up a menu of modifiable attributes for any primitive; when an attribute is selected from the menu, a dialog box allows the user to edit the attribute's value.

In the OPOSSUM prototype, model and metaphor data structures must be created by hard-coded routines. Visual editing of models and metaphors is currently being implemented. We have used the formalism to support this: a data model describing data and visual models (i.e., a meta-model) has been hard-coded. Data structures describing models and metaphors can be converted to and from schemas in the meta-model. A meta-model schema can be mapped to a visual meta-model (which visualizes models as a directed graph), and edited like any other schema. This allows any data model, visual model, or metphor to be edited (except for the meta-model itself). This demonstrates that our formalism is sufficiently expressive to describe models and metaphors.

Visualization of large schemas often presents problems. For example, the meta-model schema describing the data and visual models from our own database has approximately 1400 elements, the majority describing the visual model. The display of large schemas like this present several problems: it is difficult to see both details and gross structure, and edges in the graph stretch for long distances across the schema. Our formalism suggests several solutions to these problems, including different and more compact visual representations. For example, the metaphor could offer choice of visual primitives so that a long edge could be equivalently represented as two independent pieces. Limited space precludes a more detailed description here.

RELATED WORK

Other systems have taken a formal approach to visualization. User interface tools that are based on formal models include Chiron [6], Humanoid [5], and UIDE [4]. These provide various means for defining data and visual models. These systems, however, capture metaphors procedurally and/or as part of the visual model. Another related formal approach is that of Kuhn and Frank [2], who use algebraic mappings to examine user interface behavior. For example, they compare the behavior of a real desktop to that of a computer desktop metaphor. They do not consider data visualization, however.

CONCLUSIONS AND FUTURE WORK

In this paper, we have described an idea from the area of databases (schema mapping) applied to a problem in the area of interfaces (visualization). The result is a formalism and a tool that are beneficial to both areas. The implementation of the tool, OPOSSUM, is still in progress, but it has already demonstrated some of the power of our formal approach. Future work on OPOSSUM includes completing model and metaphor editing, enhancing the constraint language, and integrating OPOSSUM with a database system. In addition, we intend to test OPOSSUM empirically, as real users will provide the best feedback to gauge the problems and pitfalls of our approach.

REFERENCES

1. Haber, E.M., Ioannidis, Y., and Livny, M. Foundations of Visual Metaphors for Schema Display. To appear in the *Journal of Intelligent Information Systems,* Special Issue on Visual Information Systems, Summer 1994.

2. Kuhn, W., and Frank, A.U. A Formalization of Metaphors and Image-Schemas in User Interfaces. *Cognitive and Linguistic Aspects of Geographic Space,* pp. 419-434. Kluwer Academic Pub., Amsterdam, 1991.

3. Miller, R., Ioannidis, Y.E., and Ramakrishnan, R. The use of information capacity in schema integration and translation. Proc. 19th Int. VLDB Conference, pp. 120-133 Dublin, Ireland, August, 1993.

4. Sukavariaya, P. 'Noi', Foley, J.D., and Griffith, T. A Second Generation User Interface Design Environment: The Model and Runtime Architecture. In Proc. of *INTERCHI '93,* pp. 375-382, Amsterdam, April 1993.

5. Szekely, P., Luo, P., and Neches, R. Beyond Interface Builders: Model Based Interface Tools. In Proc. of *INTERCHI '93,* pp. 383-390, Amsterdam, April 1993.

6. Taylor, R.N., and Johnson, G.F. Separation of Concerns in the Chiron-1 User Interface Development and Management System. In Proc. of *INTERCHI '93,* pp. 367-374, Amsterdam, April 1993.