ESPRIT

HIFI: HYPERTEXT INTERFACE FOR INFORMATION SYSTEMS

HIFI is to create a set of tools that could be used to cheaply and quickly build a hypertext-like interface on top of an external database-information system. The interface provides interactive navigational access to information both within a database of one kind and across databases of different kinds, including multimedia. HIFI's main contributions are a model for the navigational interface, a mapping strategy to map the interface model to database structures, and the architecture of the execution tools. Three pilot applications are underway.

UMBERTO CAVALLARO
Systems and Management

FRANCA GARZOTTO and PAOLO PAOLINI
Politecnico di Milano

DOMENICO TOTARO
Systems and Management

HIFI is one of the Esprit III information-processing system and software projects. The project, which involves eight companies from three countries, started in 1992 and is to end in mid 1994. The total cost is expected to be 5.5 million ECUs (about $6.8 million). The level of effort is expected to be 39 man-years.

Although all information systems exploit a variety of media and use some kind of database technology, they usually have no mechanism to support a hypertext-like navigational style of accessing the information.

Likewise, hypertext tools provide interactive navigation, but are not conceived or structured to allow easy direct access to the information in external databases. HIFI's goal is to unite these two technologies in a front-end interface that is logically independent of the external database's structure.

RECONCILING TECHNOLOGIES

Information-system users are finding the interactive navigational style provided by hypertext tools invaluable for accessing bodies of information in different media. Tools like Hypercard, ToolBook, and Guide have become quite popular in business, education, and entertainment applications — perhaps because the navigational paradigm is so intuitive. The reader moves from one piece of information to another, following an interactive representation of links that connect pieces. However, although hypertext tools are usually conceived as managing their own information base, they lack the efficiency, robustness and variety of services provided by database systems.

Database technologies, on the other hand, suffer from a lack of support for interactive navigation, being based on different access paradigms — either queries (relational databases) or pointers (object-oriented databases).

What is needed is a hypertext-based navigational front end to a database-management system. A few tools use this approach for Structured Query Language databases, but they have no provision for restructuring the information to make navigation practical. Thus, the hypertext interface has the same structure as the underlying database schema (one card for each row of a table, with joins presented as links), making it difficult to use.

INTERFACE STRUCTURE

We felt that a better approach was an interface that was logically independent of the external database's structure. To achieve this, we designed an interface with three elements:

- High-level description of the desired navigational interface — in essence, the hypertext structure (schema).
- High-level description of the databases to be accessed — in essence, the database schema.
- Mapping that relates the two schemas.

At execution time, the system interface controls user interaction with the navigational interface. Whenever a command that requires a new piece of information is activated, the HIFI software, using the map-

0740-7459/93/1100/0048/$03.00 © IEEE

ping definition, translates it into the proper sequence of database commands.

Hypertext schema. To describe the hypertext schema (navigational interface), we created HDM+, a model based on the well-known Hypertext Design Model.[1] Both models distinguish between modeling-in-the-large and modeling-in-the-small. Figure 1 shows the two types of modeling. Modeling-in-the-large focuses on the application's structure, with no consideration of the content, which is the concern of modeling-in-the-small.

In HDM, the only concern is modeling-in-the-large. In HDM+, modeling-in-the-small is also an important ingredient. This inclusion is the most significant improvement of HDM+ over HDM.

Both types of modeling are independent of either the system to implement the navigational interface or the system to implement the database because an application is described at a logical level through a schema.

Modeling-in-the-large. The in-the-large constituents of a schema are

♦ *Entity types.* Entities are the basic objects. An entity (for example, a bank customer) is made by arranging several pieces of information in a tree structure. An entity type describes the common topological features for a class of entities (for example, all bank customers).

♦ *Web types.* Webs are the connection objects. (We based them on the web concept in Intermedia, one of the pioneer hypertext systems.) A web can connect many targets (*n*-ary connections), each being either an entity, a node-tree (belonging to an entity), or another web. A web has a center, to which a component (an in-the-small constituent) is attached. In this way, information is associated with the web. A web type describes the common topological features for a class of webs (for example, the connection between bank customers and their accounts).

♦ *Access structures.* Access structures provide ways to get information. There are two types — Index and Guided Tour — both of which are based on the existence of a list of items (for example, a list of bank customers' names). An index allows the random selection of an item from the list; a Guided Tour allows a linear scanning of the same list (starting from an arbitrary entry point).

So far, HDM+ may appear to be just another logical database model. The major difference is that it provides a navigational interpretation for the schema. The relational database model does not support navigation at all, and the OO database model provides only very crude, impractical navigation patterns based on one-way pointers.

HDM+ provides four types of navigation:

♦ *structural*, exploring the different pieces of information within the same entity;

♦ *applicative*, moving from one entity to another;

♦ *access*, navigating from access structures to other objects; and

♦ *perspective*, exploring all possible ways to view a piece of information.

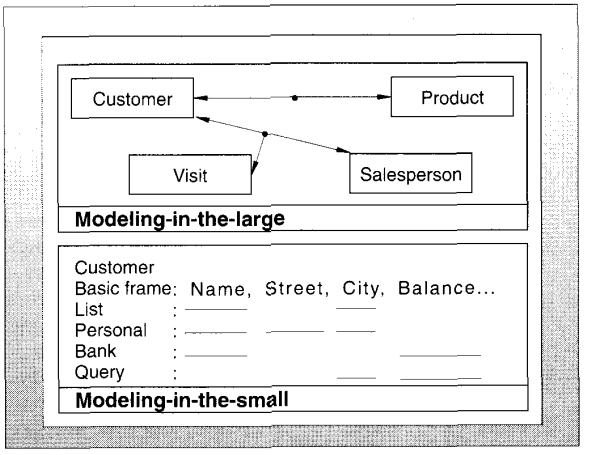


Figure 1. Modeling-in-the-small and -in-the-large.

Modeling-in-the-small. The basic element in this type of modeling is the *component*. A component contains the actual elements of the information stored in each tree node. Components let you attach information to other objects, either to a tree node or to the center of a web. A component has its own structure; it consists of several frames, each of which is an aggregate of slot values. A frame has either a hierarchical structure, because each slot value can be structured as a record, or a simple value.

Values can range from simple data to video clips, whatever types are needed. Value typing is not a concern of HDM+, but of the external database.

Each frame represents a *perspective*, a way to look at a piece of information. There are three types of perspectives.

♦ *Data perspectives.* Perspectives of this type generally contain enough data to represent the subject. Data perspectives of the entity "bank customer" would contain all the customer's personal data, data about his banking habits, and data about his family. Frames representing data perspectives are typically presented one at a time.

♦ *List perspectives.* These perspectives contain little data, instead showing several objects simultaneously in tabular format. An example is a list containing customer names and addresses.

♦ *Query perspectives.* These help define possible items that can be used in a query condition, for example, *balance* > 50 million lira and *city* = Milan.

Perspectives let designers more effectively adapt the navigational interface to user needs. They also make the transfer of data from the databases to the interface more efficient because they identify precisely what data is needed under what circumstances. This is much more economical than retrieving all the information associated with a node, which could be difficult when several databases and multimedia values are involved.

Database schema. The interface user need not know the database structure. The implementer of the interface must, however, understand the structure of the database as well as the hypertext-schema-to-database-schema mapping. We use standard SQL description for the database schema for relational databases and the database-management system's language for multimedia databases.

Mapping. The HIFI models describe the mapping between

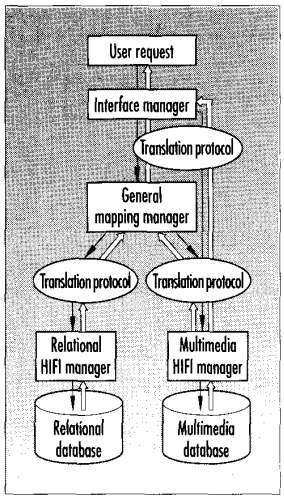


Figure 2. *The HIFI execution environment. White arrows represent data; black arrows, commands.*

the hypertext and database schemas at two levels. At the first level, it maps each slot value of each frame to a specific database manager. The current implementation allows only two managers: *R*, for a standard relational database manager, and *O*, for a multimedia OO database manager developed at the University of Crete.[2] (When a standard multimedia database is available it will simply replace the one we are using now.) We will consider other database managers in the future.

The mapping effectively partitions each frame into a number of subaggregates ("projections" in relational-database terminology), each managed by a different database manager. To ensure that it can later reconstruct the desired frame from the subaggregates, the HIFI software applies standard normalization rules during the mapping process.

At the second level are three mappings, which relate each subaggregate to the database schema: a target-list mapping, a join mapping, and a qualification mapping. The target-list mapping maps each slot into a field of the database manager. The join mapping describes the interconnections — join descriptions (for relational systems) and pointer identifications (for OO systems) — needed to reconstruct the subaggregates from the database. The qualification mapping describes possible qualification conditions on each field, which the interface may need to select the proper subset of information.

EXECUTING AN APPLICATION

Executing a HIFI application has a definition phase and an execution phase. During the definition phase, the HIFI software loads all descriptions into a dictionary. During the execution phase, it implements the navigational interface.

Definition environment. The definition environment is built around Olivetti's Tasis-Envision dictionary tool, which stores the definitions of the HDM+ specifications and the mappings. We also use several specialized editors, which are based on a reconfiguration of Tasis-Envision's general-purpose graphics editor, to insert the hypertext and database schemas and mappings.

Because the definitions must be structured in a format that the execution tools can read, we have included a downloading (off-line) facility to extract information from the dictionary to use in restructuring the definitions. Thus, execution tools do not access the dictionary directly, only the downloaded descriptions.

Execution environment. The execution environment consists of four tools:

- *Interface manager.* Manages the execution of the navigational interface. Users can either formulate queries (but only at the beginning of a session) or navigate using links and HDM+ options.
- *General mapping manager.* Coordinates the overall translation process.
- *Relational HIFI manager.* Translates requests from the general mapping manager to relational databases;
- *Multimedia HIFI manager.* Translates requests from the general mapping manager to multimedia databases.

We plan to add new managers and translation modules as new database managers are considered.

Figure 2 shows how these tools interact. The interface manager handles user interaction. For each user request that requires a new piece of information, the interface manager formulates an HDM+ request to the general mapping manager. The request is not necessary if the information was requested before and the interface manager still has the data in its buffer.

The general mapping manager translates the request into two requests, addressed to the relational HIFI manager and the multimedia HIFI manager, using a generic HDM+-to-database translation (first-level mapping).

The relational and multimedia HIFI managers then translate the generic requests into queries that are based on the database schemas (second-level mapping).

Finally, the general mapping manager gathers the database results, restructures them according to HDM+ rules, and passes the restructured results to the interface manager.

The entire interface software package is perceived by a database-management system as a normal application program, which formulates requests for query execution. Thus, there is a minimum of interference with the information system's normal operations.

APPLICATION

We are currently developing three pilot applications: sales support for a bank in Italy, support for monitoring patient conditions and medical treatment in a hospital information system in Germany, and an electronic version of the Gold of Greece exhibit, organized by the Benaki Museum in Athens in collaboration with the Dallas Museum of Art in the US. In the interest of space, we describe only the banking application.

The Banca Popolare di Brescia in northern Italy has 45 branches. Its current information system uses nine data-bases, and it plans to add a database to store simple multimedia data (such as pictures of customers and their signatures).

The main goal of the new interface is to help salespeople identify interesting customers and spot possible products to promote to them. To do this, a salesperson needs general information about customers, their families, and the companies they are involved with; information about recent transactions the customer made; information about the services the customer uses that are already provided by the bank or that could be provided; commercial information about sales activities (visits paid to the customers, outcome of the visits, and so on); and marketing information about the products and services that could be

sold to the customer.

The navigational interface must also provide a way to monitor and supervise aspects of the sales activity itself, such as salesperson productivity, product effectiveness, and market response to special offers.

A session with the interface might go something like this:

1. A marketing analyst formulates a query for customers with a deposit larger than 50 million lira, at least three children, and no credit cards.

2. The query produces a list of 10 customers, complete with name and address.

3. The list is saved under list-a and then stored.

4. Sometime later a sales supervisor picks up list-a, uses it as an index, and chooses a specific customer, whose name is Rossi.

5. The synthetic data — an abstract of all the data on Rossi — is presented, and for a while, the clerk navigates through the other information of Rossi, looking at his picture, his personal data, which services he is getting from the bank, what sales visits have been paid to him, and so on.

6. The supervisor decides that he wants to know more about the visits. The system displays a list of visits, showing the date and the name of the salesperson who visited Rossi.

7. The supervisor chooses the last visit and navigates to full information about the visit, including a short text and possibly a voice comment.

8. After reviewing this data, the supervisor has doubts about the effectiveness of the salesperson who made the visit; therefore he activates the proper link and explores additional information about him.

9. The supervisor returns to data about Rossi, writing a comment and a recommendation about potential sales that could be attempted.

10. The supervisor returns to list-a and starts another exploration.

11. Sometime later a salesperson picks up list-a and, following the comments and recommendation he finds associated to each customer, he plans his next week's sales.

These steps reflect the types of navigation possible with HDM+. Steps 1 through 4 reflect what is done in most information systems to store and access data. But step 5 shows the usefulness of access navigation. Steps 6 and 7 are examples of structural navigation. Step 8 and 9 are examples of applicative navigation.

The objectives of HIFI have been verified in a number of meetings with potential users, both those who participated in the pilot applications and those who obtained project information from papers or seminars. Most potential users we interviewed were frustrated with the difficulties of getting hands-on control of the information in databases and viewed the new interface with great hope.

After one year, we have fully specified HDM+ and developed techniques for mapping. We have defined the architectural details of the interface, done a detailed design of each execution tool's internal structure, and specified the communication protocols for the tools. Finally, we have run experiments to verify the overall feasibility of the HIFI approach and begun implementing the execution tools.

In the remaining year, we plan to complete tool implementation (around March 1994) and finalize the pilot applications. We also plan to start up other applications as follow-up work for ongoing commercial contracts. ♦

ACKNOWLEDGMENTS

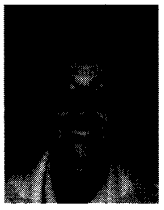
The organizations involved in HIFI are Systems and Management of Italy (coordinating partner), Epsilon of Greece, Siemens of Germany, Syntax of Italy, Benaki Museum of Greece, GMD/IPSI of Germany, Music/Forth of Greece, and Politecnico di Milano of Italy.

REFERENCES

1. F. Garzotto, P. Paolini, and D. Schwabe, "HDM: A Model-Based Approach to Hypermedia Application Design," *ACM Trans. Information Systems*, Jan. 1993, pp. 1-26.
2. S. Christodoulakis et al., "An Object-Oriented Architecture for Multimedia Information Systems," *IEEE Data Eng. Bulletin*, Sept. 1991.

Umberto Cavallaro is coordinator of research and development projects at Systems and Management, where he collaborates in large intercompany projects, involving hypertext/multimedia systems and end-user computing applications. He is also the HIFI project manager.

Cavallaro received a PhD in philosophy from the University of Turin. He is a member of the International Standards Organization.



Franca Garzotto is a senior researcher in electronics and information at Politecnico di Milano. Her research interests are database constraints, software specifications, conceptual modeling of documents, hypertext and hypermedia modeling, hypermedia authoring systems, and multimedia development tools.

Garzotto received a PhD in computer science from the Politecnico di Milano. She is a member of the IEEE and ACM.

Paolo Paolini is an associate professor in electronics and information at the Politecnico di Milano, where he manages the multimedia laboratory. He has also been technically responsible for several ESPRIT research projects. His interests include database modeling and systems, languages, distributed databases, database views, hypertext and hypermedia modeling, hypermedia authoring systems, and multimedia development tools.

Paolini received an MS and a PhD in computer science from the University of California at Los Angeles. He is a member of the IEEE and ACM.

Domenico Totaro is a senior consultant at Systems and Management, where he has been involved in projects involving information-systems modeling, hypermedia authoring systems, and database interfacing. He is also involved in the design and implementation of information systems for banking.

Address questions about this summary or requests for HIFI documents to Cavallaro at Systems and Management, Via Alfieri, 19, 10121 Torino, Italy; Internet umberto@to.sem.it.