Payoff

One of the biggest challenges facing organizations is providing support, education, and consistent workable systems disciplines for users developing complex information systems. Without this support, users run a risk of repeating mistakes well known to IS, including development backlogs and major information systems that are poorly designed and difficult to integrate with other systems. A potential solution for these problems is an end-user systems development environment that employs expert systems technology. This article describes a prototype intelligent development support system for end users.

Introduction

Disappointing development team productivity, cost overruns, unsatisfactory quality, and systems that are difficult to maintain are examples of software crises. A major factor contributing to software crises is the complexity of new systems developed to address ever more ambitious business challenges. Users, too, have greater expectations for enhanced technological support.

Solutions to software crises fall into two general categories. The first gives systems developers powerful tools that help them deliver more effective, efficient service. Software development tools include various computer-assisted software engineering (CASE) systems, automatic data dictionary, reusable code, structured design and programming, integrated data base management and knowledge base management components, and executable requirements specification languages.

Another approach focuses on the user's role in the systems development task. Users of information systems can have a meaningful involvement in the design of their own applications. Users can also share the responsibility for maintenance of the applications they develop. This strategy not only reduces the backlog of new applications, it leads to a lighter maintenance load for the IS or EUC department, allowing more professional development time and effort to be directed to other critical corporate goals.

Achieving the Benefits of End-User Computing

The positive impact of end-user applications development can be realized only if users are provided with a conducive development environment and if systems integration problems are resolved. A basic premise of this article is that these two constraints on end-user computing can be handled as two different aspects of the same problem. Essentially, the objective is to develop an intelligent development environment that supports users and eliminates integration problems. Concepts from software engineering and artificial intelligence that form the basis of this development environment are:

- The data-centered systems development methodology.
- The Information Resource Dictionary System.
- The object-oriented paradigm for information modeling.
Integrating User-Developed Applications

Data base technology has been widely recognized as an effective means of integrating applications developed by users. Centralized data management promotes data sharing and results in applications developed with interfaces prescribed by the data model. An Integrated Data Dictionary facilitates functional communication among system components; an Information Resource Dictionary System contains information about all types of information components, including data, hardware, input screens, and report format definitions. Both concepts were instrumental in the design of the architecture of this intelligent development environment.

The object-oriented paradigm was selected because it provides a convenient mechanism for modeling heterogeneous types of objects. In this system, objects are used as the only knowledge representation scheme for all the interface constructs as well as the data bases.

Data-Centered Systems Development

The data-centered approach to systems development is characterized by applications built on a predefined data base structure. It was adopted in this project to resolve integration and integrity problems. The data-centered systems development methodology offers the following advantages:

- It avoids the problems of file proliferation, maintenance, data redundancy, and inconsistency.
- Once appropriate data bases are in place, some types of applications can be created quickly with high-level data base languages.
- Users with direct access to data bases can create their own reports and applications, often without the slow steps of formal systems analysis and without waiting for the IS or EUC department.

The logical structure of a data base captures important data semantics from the user's point of view. Users access data under coordination of the data base management system. The multiple-level schema mapping mechanism of the DBMS provides data manipulation entities (i.e., application program and terminal users) with easy-to-use integrated interfaces. The degree to which a DBMS succeeds in integrating users' data manipulation activities is significantly affected by the quality of logical data base design.

Designing a logical data base entails three tasks: data item definition, intraentity and interentity relationship identification, and normalization. Data definition and relationship identification are basically context-sensitive; normalization is essentially mechanical. Although users generally understand the meanings of the data they work with, very often they have only limited design knowledge. A user—oriented assistance system for data base applications development must maximize support for normalization and minimize prompting for data definition and relationships specification.

Integrated Information Resource Dictionary

A traditional data dictionary contains definitions of data items and serves as a passive information source. Separated from the other components of the DBMS, a passive data dictionary represents a standalone repository of data definitions—basically it serves as a
documentation tool. In contrast, an integrated data dictionary (IDD) constitutes an active element of the data base management environment—that is, active links allow direct communication with the operational components of the data base system. This communication enables various types of integrity constraints to be automatically enforced when the user performs data manipulation.

An information resource dictionary system is a data dictionary system expanded to include a wider range of information resources. In addition to data items, an information resource dictionary system also contains information about programs, users, hardware, and decision models. For all practical purposes, however, the range of information resources must be determined by the user's current and future requirements. For example, in the context of user applications development, the entities that should be stored in an information resource dictionary system include data entry screens and report formats as well as data item definitions. Data entry screens and report formats stored in the information resource dictionary system can be shared by anybody having access to the system. An integrated information resource dictionary system was used as a central element in this support system to serve as a central communication component in the system, functioning as a link between the systems development phase and the systems operation phase.

Object-Oriented Paradigm

Object-oriented concepts are often favored for information systems development in general and information modeling in particular. In this paradigm, real-world entities, together with their behaviors, are represented by objects. Objects communicate with one another by exchanging messages. A message is actually the kind of operation or procedure to be performed.

An object is made up of three generic components: a data structure to describe unique characteristics of the object itself, a specification of the actions to be performed by or on the object, and a message-acceptance protocol. After it is created, an object can be invoked without full knowledge of its detailed internal structure (a feature referred to as encapsulation). Representing entities as objects has the potential of reducing much of the complexity of integrating different types of data. The reusability of the object specifications further improves the productivity of systems development.

Another important aspect of the object-oriented paradigm is hierarchy. Properties of objects at a higher level are inherited by objects at lower levels. Property inheritance not only helps accomplish storage economy but makes integrity constraints easier to enforce, an important requirement for an end-user computing support system.

An object-oriented system is relatively easy to maintain. Because every object is separate from any other object, modification of one object does not necessitate revision of the other objects; therefore, objects can be added and deleted with minimal ripple effect throughout the system. This feature is one of the chief reasons the object-oriented approach appeals to many systems developers.

The object-oriented paradigm can provide important conceptual as well as operational guidance throughout the end-user computing support system's development process. All logical entities are modeled and implemented as objects. The major benefits of the object-oriented paradigm are:

- **It provides a uniform modeling vehicle for different types of entities.** The knowledge required for data base normalization can be represented as objects and implemented in the same fashion as other entities.
It permits experimental prototyping. In this case, a complex system was constructed and refined gradually as knowledge of functional requirements accumulated over time. Even a major change of system requirements did not necessitate a complete rewriting of the entire system.

The Prototype System's Architecture and Operation

To support end-user applications development, the system must provide intelligent assistance throughout the systems development and operation process. During the applications development phase, the system automatically performs normalization on the basis of the user's selection of candidate keys. Knowledge about keys and functional dependencies is domain-specific and entirely a semantic issue. Because users typically know more about their business than IS professionals do, it is important to provide a user-friendly interface—in this case, an interactive, question-answer interface.

The system helps the user set up screen layouts and report formats. The data base structures, screen layouts, and report formats are all stored in a dictionary and are used by the system in the operation phase (see Exhibit 1).

System Architecture

During the operation stage, users perform data manipulation activities (e.g., record additions and deletions, updates, and report generation) with the system's assistance. From the user's perspective, a data base system is not much more than a collection of screen layouts, menus, and reports. Both the physical and the logical structures of the data bases can be hidden from the user without impeding the user's data manipulation tasks. The different interface constructs (e.g., menus, screen forms, and reports) are treated as objects, with transitions between objects accomplished by sending messages (see Exhibit 2).

System Implementation of the User's View

Transparent to Users

From the user, the system receives definitions of data items and identification of candidate keys associated with a specific task domain. The system automatically derives a set of third normal form relations from these user specifications. The system interactively assists the user in setting up report formats, screen layouts, and menu options. The resulting specifications constitute an initial prototype that the user can verify, revise, and extend.

In this system, all the internal representations of interface constructs (e.g., reports, screens, and menus) and data bases are transparent to the user. A user view generator incorporates data items from several data bases. The system stores all of these in the Object Specification Dictionary (an Information Resource Dictionary System). The user can iteratively refine the prototype until it becomes an operational system. Exhibit 3 depicts the applications development process using this system.
The User Applications Development Process

The functional components of the user-oriented system—that is, the part of the system that handles the applications development stage—fall into two levels. At the top are the menu reviser, the screen reviser, the report reviser, and the data base designer and developer. These program objects collect domain-specific information from the user. They also call appropriate modules at the second level to generate object specifications. These object specifications are kept in the object specification dictionary for use at the systems operation stage. The logical relationships of these modules are shown in Exhibit 4.

Functional Components of the Intelligent Development Aid

Conclusion

IS must enable business users to get the most value from information technology. However, the benefits of end-user computing can be realized only if user-developed applications are integrated into data bases of integrity.

Toward this end, IS should create conducive development environments and development aids to coordinate business users' efforts in building and using applications. This article has presented some ideas on the design of a user-oriented assistance system for data base applications development. The basic assumption made in selecting the foundation technologies, formulating the system architecture, and implementing the prototype is that users need maximum support from the applications development tools they use. Another objective was to create an applications development environment that eliminates integration problems.

One drawback of the prototype is the lengthy response times of most of the screens; however, the interactive interfaces couple the users' domain knowledge with the system's knowledge about data base applications design and implementation. The outcome is an example of an easy-to-use, intelligent applications development and operation environment for business users.

Author Biographies

W.J. Kenny Jih

W.J. Kenny Jih is an associate professor in the department of management at the University of Tennessee, Chattanooga. His research in strategic applications of information technology, expert systems development, data base design, and data communications has been published in several international journals.
Output
- Object Specification
- Dictionary

Procedure
1. Input Forms and Normalization
2. Specify End-User Database
3. Specify Report Process
4. Specify Display Screen Process
5. Specify Menu Interface
6. Initial Prototype

Steps:
- Operational System
- Working Prototype
- Is the user developer satisfied?
- Can it be revised or enhanced?
- Yes
- No