

# Leveraging Internet And Database Technologies To Enhance Internal Business Processes

Thomas G. Calderon, (Tcalderon@uakron.edu), University of Akron  
David H. Olsen, (dolsen@uakron.edu), University of Akron

## Abstract

*Many small companies use the Internet for advertising, browsing, and electronic mail. Few companies attempt to combine Internet technologies with their database management systems in order to manage and control their routine work processes. This paper discusses a case in which a small, fast growing service company enhanced its work flow, reduced its cycle time, reduced paperwork and improved employee morale by combining Internet and database technologies. The paper also describes how the company subsequently made significant improvements in the system through a focused goal-oriented review. The benefits and processes used in such a review are discussed.*

## Introduction

Many accounting firms and other service enterprises organize business activities into sequential tasks that are completed by employees working individually or in groups. An individual performs one or more tasks and then passes the completed task(s) to another person who completes the next task(s) in the sequence. This process continues in a predetermined sequence until all tasks are completed. When groups are cohesive and group members are within close proximity to each other, work progresses effectively, and activities are usually completed on time (Vinokur-Kaplan, 1995; Klein, 1995). However, as the group grows or becomes more dispersed, task coordination and

supervision become major problems.

Companies can use a wide array of technologies, ranging from basic electronic mail applications to sophisticated group decision support systems (GDSS), to coordinate and manage group activities. Although sophisticated GDSS applications (e.g., GroupSystems, TeamFocus, VisionQuest, and Lotus Notes) are generally effective, they are often costly to purchase and must be customized in order to support the group activities of most service enterprises (Bidgoli, 1996). Some of the more commonly used commercial GDSS applications range in price from \$15,000 to \$75,000. Basic electronic mail applications, on the other hand, are inexpensive and firms of all sizes use them for communicating group and individual activities. Electronic mail

---

*Readers with comments or questions are encouraged to contact the authors via e-mail.*

provides business professionals with user-friendly, convenient, and fast global access to internal and external constituencies. Short notes, large documents, and complete files can be transmitted over the Internet using e-mail technology. E-mail is usually one of the most efficient forms of business communication when entities are separated by long distances, when written documents must be exchanged, and there is a need for accurate archives (Wheeler, 1995). This makes e-mail a highly attractive business communication tool and places the technology at the top of the list of available communication methods.

Despite the versatility of basic e-mail, it is not usually coupled with database technology to provide a system for coordinating and managing routine business processes. This paper presents a case that shows how a growing service company combined e-mail and database technologies to build a cost-effective system for coordinating and managing group activities, and enhanced many of its internal business processes. The paper also discusses how the company subsequently improved the system through a goal-oriented technology review process.

## **Background**

ASC Inc.<sup>1</sup> is a fast growing company that provides retirement plan management and administration services to more than 770 clients nationwide. Many business and not-for-profit entities outsource their retirement benefits administration and other related human resources activities to ASC. The company performs several major activities in providing service to its clients. Each major activity involves distinct tasks performed by employees in different departments. For example, eight specific tasks, involving eight different employees, are required to add a new investment to a retirement plan. Tasks performed by different departments are centrally coordinated and monitored to ensure that the company provides a high quality, error-free service to its clients.

Before December 1995, ASC used a manual system to execute, coordinate, and supervise its activities and tasks. A task completion form was used to manually track the progress of activities and tasks through the system. As each task was completed, the employee involved checked a box on the form to indicate its completion. The form and corresponding client documentation were then forwarded to the employee responsible for completing the next task. Completion of a single activity required performance of many tasks spread over up to eight departments. Because the task completion form and the related documentation might be in the hands of an employee in any one of those departments at a given time, coordinating and monitoring the progress of activities and tasks were major problems. Keeping track of client files was also becoming a major concern. Although the company seldom experienced situations where a required task was not completed on schedule, management realized that this type of failure could cause major operational problems and expose the company to significant losses.

The manual system served ASC adequately while the company was small. However, as ASC grew, a considerable amount of employee time was spent retrieving, processing and summarizing task completion forms. Management recognized that the manual system exposed them to several potential problems. Three areas of risk concerned the company. First, the task completion checklist and the associated client files could be anywhere in the organization and it was possible that no one would know the overall status of a particular activity. Potential consequences include work delays, inability to service customer calls, and financial loss to the company and its clients. Second, persons responsible for completing a particular task would not know when the task that preceded theirs was completed. Therefore, the progress of tasks through the system could be slowed and activities would not be completed in a timely fashion. Because timeliness is a necessity in the financial services industry, this was considered to be a major risk. Third, departments responsible for delaying the

completion of an activity were not clearly identified and attempts at corrective action could be focused at the wrong department or individuals. This could lead to morale problems and, ultimately, excessive employee turnover. To avoid these potential problems, the company determined it needed a system that would: (1.) track the tasks to be completed and identify persons responsible for various tasks as an activity is executed; (2.) keep track of client files used in the completion of activities; (3.) provide timely advice on when new tasks and activities are completed or need to be started; (4.) reinforce the need to complete tasks and activities in a timely fashion; (5.) provide timely feedback to employees, supervisors, and managers; and (6.) streamline the coordination and supervision of activities; and (7.) reduce paperwork. Overall, ASC needed a system that would streamline, track and audit its work flow. This system would be designed to reduce the risk of work delays, inability to service client inquiries, and financial loss to the company and its clients.

The idea for a solution to the problem originated from a brain storming session between the vice president for service quality and employees in the MIS department. The idea involved the marriage of e-mail technology with database technology. Both technologies were already being used by the company but, as in many other organizations, no one had thought of combining them to solve the problems listed above. ASC designed, built and implemented a low cost system that coupled e-mail and database technologies in approximately one month. The progress of all tasks throughout the company was easily tracked and reviewed through this system, leading employees to

refer to it as the *Enforcer*. Software used in the system included: (1.) a mail server that distributed e-mail to department supervisors and employees; (2.) Microsoft's SQL Server that functioned as the main database engine for initiating e-mail, feedback and reports, and for maintaining a complete history of tasks and activities for all clients; and (3.) Microsoft Access coupled with Visual Basic to design and build the reports and forms that served as the main user interface.

### The Original Version of The Enforcer

The *Enforcer*, initially implemented in December 1995, sent e-mail to employees and supervisors daily to inform them of new activities and tasks. The status of all activities and tasks handled by employees were also summarized in the daily e-mail sent to supervisors. The e-mail included an electronic task completion form, designed using Microsoft's Access, that replicated the form used in the original manual system. Figure 1 shows an outline of a form used for one of the company's activities, a Daily Valuation Fund Change. Among other things, the form identified all tasks required to be complete an activity and showed the status of each

Figure 1  
An Outline of a Form Used in the *Enforcer*

Client:  Administrator:  Change effective:

Change is an addition of a new fund

Change in fund name only

Change is a removal of an existing fund

Description:

Required	Completed	
<input type="checkbox"/>	<input type="checkbox"/>	Description of task 1
<input type="checkbox"/>	<input type="checkbox"/>	Description of task 2
<input type="checkbox"/>	<input type="checkbox"/>	Description of task 3

task at the time the e-mail was sent.

In general, the system allowed the company to track and log all tasks and activities assigned to its employees. The system triggered mail and directed it to the responsible employees whenever a new activity was initiated. Each activity consisted of several tasks that had to be completed in order to complete the activity. When a task was finished, the responsible employee would place an "X" in the completed column of the form to show its status (complete). To advise employees of the tasks they had not completed, the system sent mail automatically to employees, copied to their supervisors, twice weekly. Thus, the system continuously reminded employees of tasks they needed to complete while providing this same information to their supervisors.

To deal with the problem of tracking client files, ASC implemented an electronic system for requesting, logging-in and logging-out client files. Files were requested in person or electronically from a remote location. An inventory of client files was maintained on a database, and all client files were bar-coded and kept in a file room staffed by an employee. Each file was scanned before it left the file room and data about the employee requesting the file were entered into the computer manually or scanned from the employee's identification card when files were requested in person. The files were again scanned when returned to the file-room and the user was relieved of responsibility for them. This process ensured that all client files were efficiently tracked and accounted for. It also reduced file retrieval time, helped to streamline the workflow, and increased employee productivity.

### **Improving The Enforcer Through a Goal-oriented Review**

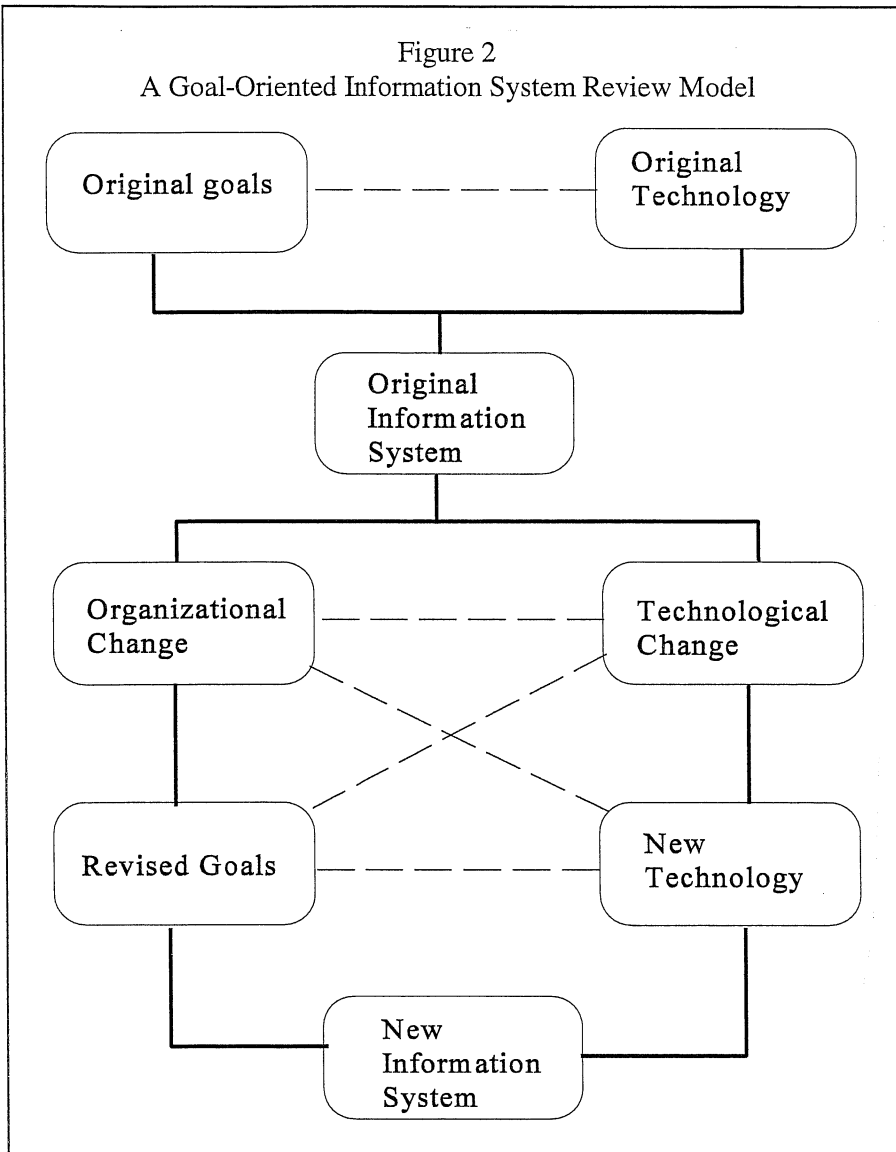
Subsequent to the initial implementation of the *Enforcer*, neither the technology used in the system nor the organization remained static. Although the *Enforcer* was initially very effec-

tive in accomplishing the basic objectives for which the system was created, the company continued its rapid growth in 1996, and as IS personnel, managers, supervisors and employees worked with the system, they observed several areas that needed improvement. E-mail technology and the methods used to interface e-mail and database technology were also changing.

As shown in Figure 2, changes in an organization coupled with changes in technology often necessitate a review of the information systems used to manage the unit's internal processes. Changes in an organization usually create new information management problems while changes in technology imply that the organization can capitalize on new or improved opportunities for solving these problems. In the case of ASC, Inc., through a combination of industry growth and acquisition activity, both the number of employees and the number of clients more than doubled shortly after the *Enforcer* was implemented. Initially, the *Enforcer* was designed for neither a sizable staff nor a large client base. At the same time, e-mail technology and protocols governing connectivity between database systems and Internet related technologies were constantly being improved. Thus, ASC undertook a focused, goal-oriented review of the original *Enforcer*. Some of the problems encountered and the proposed improvements are described in this section.

### **Goal-Oriented Review**

One of the first problems encountered in the original version of the *Enforcer* related to the link or relationship between the tasks required to complete an activity and the employees responsible for each task. At the time the *Enforcer* was originally designed, ASC was relatively small. Therefore, the relationship between employees and tasks was hard coded into the program instead of being designed into the database as a separate table that captured the dynamic relationship between employees and tasks. This meant that every time an employee left the company or transferred to a new position, or a task was



transferred to a different department, the main *Enforcer* program had to be modified and recompiled to fit these changes. Though this is a classic update problem associated with file-oriented application programs (Robb and Coronel, 1997), it was not considered to be a major obstacle at the time the system was designed because the company was relatively small, had little employee turnover, and experienced very few changes in the employee-task relationship. However, the company almost doubled in size since the original *Enforcer* was implemented in 1995 and the relation between tasks and employees emerged as a major problem that had to be resolved. Addressing this problem became a primary goal for the revised version of *Enforcer*.

A fundamental problem related to the way the relation between tasks and employees was implemented in the original *Enforcer* was that it restricted the number and type of tasks that could be handled by the system. Only a fixed set of tasks that existed or were anticipated at the time the *Enforcer* was originally designed was programmed into the system. However, as the company grew and the nature of its business changed, employees had to perform several new tasks in order to complete a given activity. Each time a new task was added, the original *Enforcer* had to be recompiled and the database needed to be altered to support tracking and reporting on the new task. This was a complex operation because the completion logic of the program also had to be altered to account for the

new task. ASC needed a more flexible database design and a more adaptable platform to access, retrieve and display data to users. This was the second goal of the new *Enforcer*.

Finally, as ASC grew, its information system became more diverse in terms of the computer environment, including system configurations and operating systems. Version control emerged as an issue. The company operated a single version of the original *Enforcer* on a network server. IS personnel had to insure that each computer was carefully and distinctively configured to access the SQL databases, and also had all the supporting Visual Basic dynamic link

libraries (DLLs). (A DLL is a collection of executable procedures, grouped to one file on a disk, that can be shared at run time by multiple Windows applications.) This maintenance effort consumed excessive resources while maintaining a status quo that was increasingly unsatisfactory. Version control and hardware compatibility are software maintenance problems that extend beyond the *Enforcer*. The company decided to deal with these problems by proactively embracing platform-independent systems--applications that can function under different computer operating systems.

### The Revised Enforcer

As a result of the company's comprehensive goal-oriented technology review process, ASC's information systems team decided to revise the original *Enforcer*. The review resulted in the following three main goals for an enhanced version of the *Enforcer*: (1.) insure that the system can handle the dynamic relationship between employees and tasks that had emerged as the company grew; (2.) accommodate the increasingly diverse and numerous tasks that the company had to complete to service its many clients; and (3.) develop a platform-independent system to handle the complex range of hardware and operating systems that evolved as the company grew. Procedures used by the company for enhancing the system are described below. These procedures are continuously being reviewed and examined by the company. Like the technology in this area, the *Enforcer* is constantly evolving and the material presented below represent one version of the system as it moved through this evolutionary process.

#### *A new employee database system*

One of the first, and most essential, steps in creating the new version of the *Enforcer* involved designing a database system that supported a dynamic relationship between employees and tasks. When the original version of the *Enforcer* was developed, the company did not have a comprehensive or "main" employee data-

base. There were several databases that departments used to track employee information but no one system was viewed or designed as "the" employee database. The first step was to design and build this database. The company was then able to construct a comprehensive database for the new *Enforcer* that used the employee database to create the dynamic relationship between employees and tasks. Figure 3 shows the design of this new employee database.

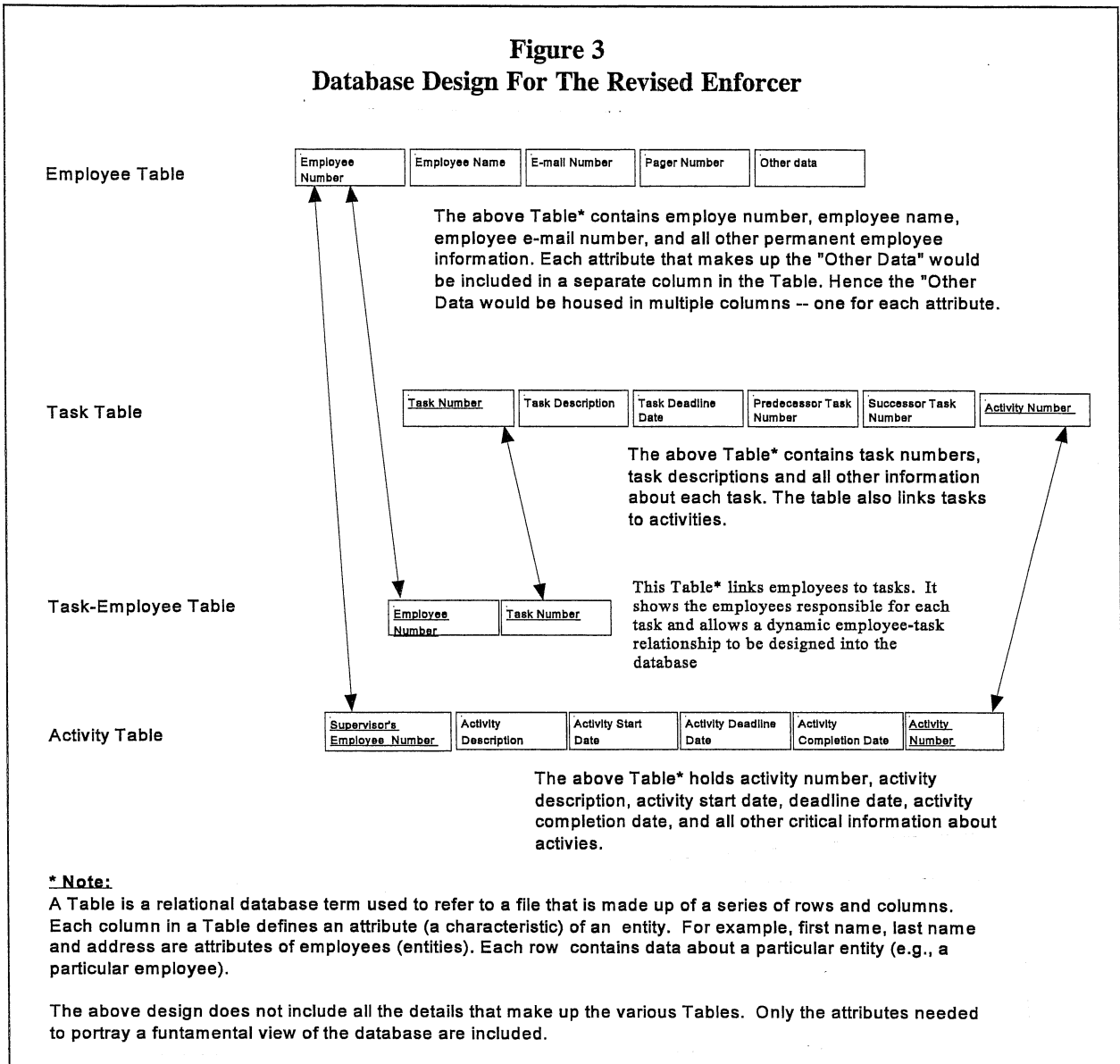
#### *A new task database*

Once the employee database was designed, the next step was to create a comprehensive task database that would allow IS personnel to remove the original set of tasks from the initial version of the *Enforcer* program and create the foundation for a more dynamic structure. With the new structure, the list of tasks can be updated and the link between employees and tasks created at any time without having to modify the main *Enforcer* program. The task database and the link between employees and tasks (see Figure 3) can now be updated on demand by an administrative assistant instead of a skilled IS employee. In addition, each task can be dynamically linked to an object that describes the task in detail and tells employees how it should be completed. Dynamic links to task details facilitate training and help ease the transition of new employees into the company.

#### *Enhanced Database Connectivity*

To provide for platform-independence, the revised *Enforcer* uses a Web interface and runs over the company's intranet. All computer systems at the company are equipped with a Web browser that allows users to link to the mail server and the SQL server. By leveraging Internet technology, maintenance of all databases are done through a series of Web-forms executed from a browser. This has turned out to be a very effective approach for dealing with version and hardware compatibility problems. The Web browser serves as a universal thin client that can execute a variety of programs despite the client

**Figure 3**  
**Database Design For The Revised Enforcer**



hardware and operating system.

The revised *Enforcer* allows employees to access the SQL database directly from a Web page. Forms and responses to queries are dynamically returned to their HTML browsers. The system uses an Open Database Connectivity (ODBC) interface to link the database system to a Web browser. ODBC interfaces allow users to work seamlessly with data that reside in many different types of database management systems. With ODBC, a company can store data in a wide variety of databases on different types of platforms, and then access the data through a single, standard programming interface. These interface

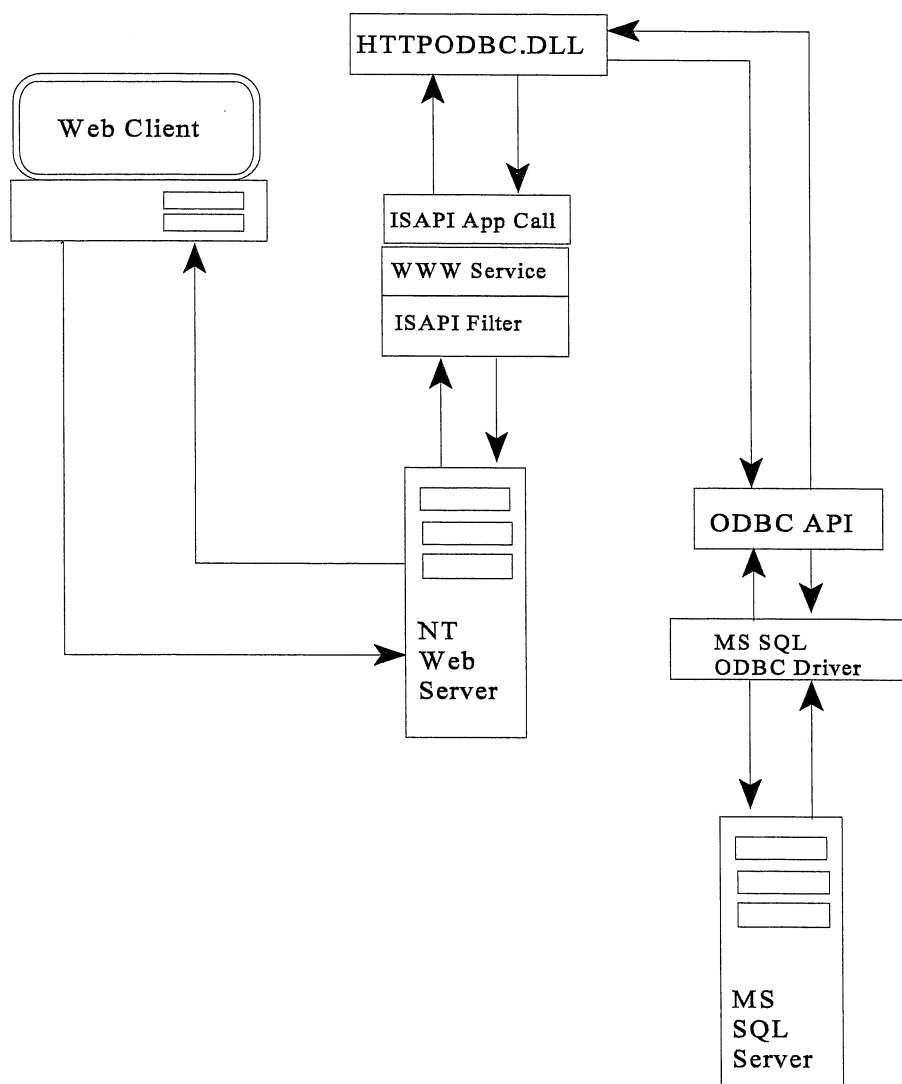
programs, once accessed, make it possible for users of all types of computers to access and work with any database management system that is equipped with an ODBC driver. In essence, ODBC application interface programs add platform independence to proprietary database management systems (Sippl, 1995). To date every major vendor of database management systems--including, Oracle, Sybase, IBM, Borland, and Microsoft--has added ODBC support to its products.

Details involved in presenting information from an ODBC datasource to a Web client using the Internet Database Connectivity (IDC)

package that is bundled with Microsoft Internet Information Server (IIS) are described below. These details provide a general overview of how the revised *Enforcer* incorporates Web connectivity. Figure 4 presents a visual overview of the system.

IDC is implemented through an Information Server Application Programming Interface (ISAPI) program called HTTPODBC.DLL. This program is incorporated into IIS and actually becomes part of IIS. An ISAPI filter is created that sends database requests to the

Figure 4  
General Overview of Web Connectivity For The Revised Enforcer



**Notes:**

HTTPODBC.DLL is an Internet Server Application Programming Interface (ISAPI) that allows a database management system with an Open Database Connectivity (ODBC) driver to be accessed, read, and updated using a Web browser installed on any type of end-user computer platform (Web client). The link between the HTTPODBC.DLL program, installed on the Web server, and the ODBC driver is established through an ODBC Application Program Interface (ODBC API) located on the main database server (MS SQL Server).



HTTPODBC.DLL for processing. Database requests from a web client are triggered by a GET or POST statement as illustrated in the following HTML code:

```
<HTML>
<HEAD>..</HEAD>
<BODY>
<FORM METHOD = POST ACTION =
  "/appc/idc" >
Enter a name to search for:
<INPUT NAME="FIELD1" TYPE=TEXT>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

When the user submits this form, the ISAPI filter catches all files that have an .idc extension and the WWW Service sends these files to HTTPODBC.DLL for processing. The following is an example of an .idc file that could be associated with the above HTML:

```
Datasource: SampleData
Username: Guest
Password: Guest
Template: sample.htx
SQLStatement: +select name, address, phone
from employees where name like
' <%FIELD1%> %'
```

This file can be interpreted as follows: the Datasource informs HTTPODBC to use the SampleData DSN (data source name) as the target ODBC data source for processing the request. Username and Password are account information fields for accessing the data in the SampleData DSN. The template is a Hypertext Markup Language Extension File (.htx) (see below) that is used to format the results of the request. The SQL Statement is a SQL statement that is executed by the database server or engine.

The .htx file is an HTML file that contains special tags that are interpreted by HTTPODBC. These tags tell the HTTPODBC program where to insert the results of the SQL

query into the HTML form to create a document that contains results of the SQL statement. The following code shows a sample .htx file that could be associated with the sample .idc file.

```
<HTML>
<HEAD>?</HEAD>
<BODY>
<%BEGINDETAIL%>
<%IF CURRENTRECORD EQ 0%>
Results of searching for <%idc.Field1%>
<TABLE BORDER>
<TR> <TH>NAME<TH>ADDRESS<TH>
  >PHONE
<%ENDIF%>
<TR> <TO> <%name%> <TD> <%address%> <TD> <%phone%>
<%ENDDetail%>
<%IF CURRENTRECORD EQ 0%>
Sorry, no records are like <%idc.FIELD1%>
<%ELSE%>
</Table>
<%ENDIF%>
</BODY>
</HTML>
```

IDC tags are denoted in the above code by <% and %> rather than the tags (< and >) used by regular HTML (e.g., <HEAD > and </HEAD>). The statement <%begindetail%> tells the HTTPODBC where to begin inserting records and <%enddetail%> indicates to the HTTPODBC where to end inserting records. Results for the code between the <%begindetail%> and <%enddetail%> tags are sent back to the client for each record encountered in the database. The system pulls fields into the detail section by enclosing field names in <% and %>. Thus, to make the "phone" field appear in a table, the system uses the tag <%phone%>. It is also possible to display the parameters that were sent to the IDC file by prefixing the parameters with idc. For example, the code listed above displays the parameter specified for searching by using the <%idc.FIELD1%> tag. FIELD1 represents the search parameter.

IDC programs may also include “IF . . . THEN” condition statements by using `< %if% >`, `< %else% >` and `< %endif% >` tags. These are used to conditionally return lines to the web client. The program can test for no records returned and/or the beginning of a record set by using the tags for the IF . . . THEN condition statements and the CURENTRECORD system variable. CURENTRECORD begins with 0 and is incremented each time the program loops through the detail section. If no records are returned then the detail section is skipped. So, if after the detail section CURENTRECORD is tested and it still shows up as 0, then it implies that no records were returned. A general template for an .htx file is presented below to clarify the use of the IF . . . THEN tags and the CURENTRECORD system variable.

```
<!--static HTML --!>
< %begindetail% >
< %if currentrecord eq 0% >
<!--table header information and other header in-
      formation goes here (row names etc.)--!>
< %endif% >
<!--record formatting is done here, table rows
      or other names of showing data--!>
< %if currentrecord eq 0% >
<!--What to show if no records were returned --
      !>
< %else% >
<!--close out tables and other structures--!>
< %endif% >
<!--static HTML--!>
```

In summary, the net result is that the HTTPODBC executes the SQL Statement supplied in the .idc file, takes the result set (if any) and combines it with the specified .htx file to produce a dynamic HTML file that is sent back to the web client. Though not illustrated above, IDC can also execute INSERT and UPDATE statements along with stored procedures, implying that IDC covers all the necessary actions to implement the data handling of the *Enforcer*. The company is in the process of building other ODBC interface application programs to make many of its other database applications platform-

independent.

### *Use of advanced database technologies*

The revised *Enforcer* makes use of several advanced technologies. One of these is active databases which use *event-condition-action (ECA)* rules (also referred to as *production rules*) to augment an otherwise passive database (Brownston, Farrell, Kant, and Martin, 1985; Dayal, Hanson, and Widom, 1995; Widom and Ceri, 1996). These *production rules* have their origin in the fields of artificial intelligence and expert systems rule languages. Rules are executed when a condition in the data triggers an action. Consider the following example from a manufacturing system:

```
On Event:      replace to PARTS.Cost
If Condition:  PARTS.Name = "3/4 screw"
Then Action:   replace PARTS (Cost =
               new.Cost) where PARTS.Name = "9/16
               screw"
```

This rule instructs the database management system (DBMS) to watch for the event where the cost of a 3/4 inch screw is updated. When this event occurs, the DBMS is instructed to perform the corresponding action which is to change the cost of the 9/16 inch screw to be the same as the 3/4 inch screw. Another rule could be defined that updates the cost of a 7/16 inch screw whenever the cost of a 9/16 inch screw is updated. A rule that updates one record which then updates another record and so on is referred to as forward chaining. When there are no more rules to process, the chaining discontinues. In this example, the rule is applied and the active database executes the appropriate action.

When applied to the *Enforcer* and the database design shown in Figure 3, an example of a production rule would be as follows:

```
On Event:      Insert to TASK-EMPLOYEE
If Condition:  TASK-EMPLOYEE.Task Num-
               ber = "10"
Then Action:   Insert Task-Employee (Task
```

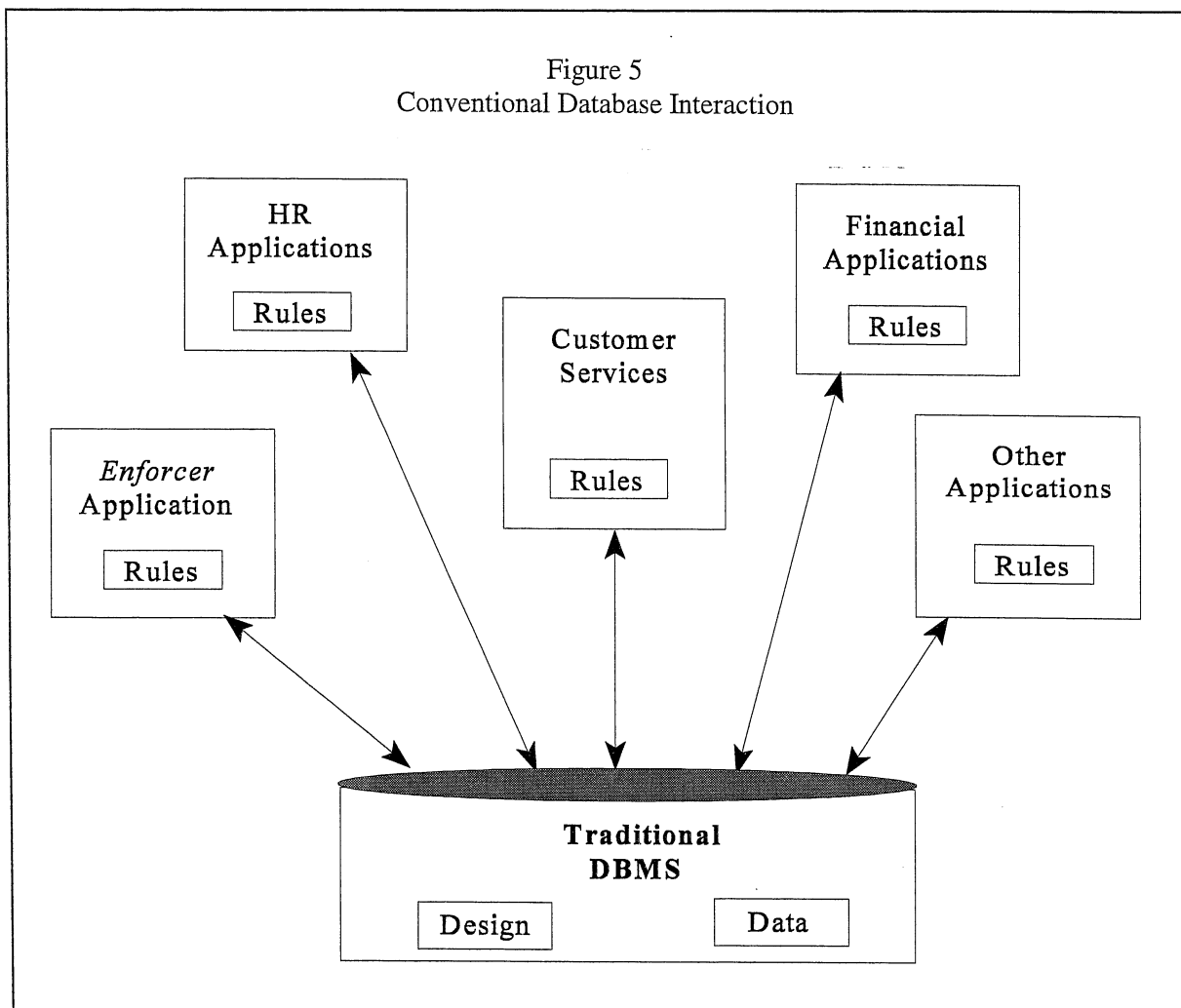
Number = new.Task Number) where  
 TASK-EMPLOYEE.Task Number = "11"

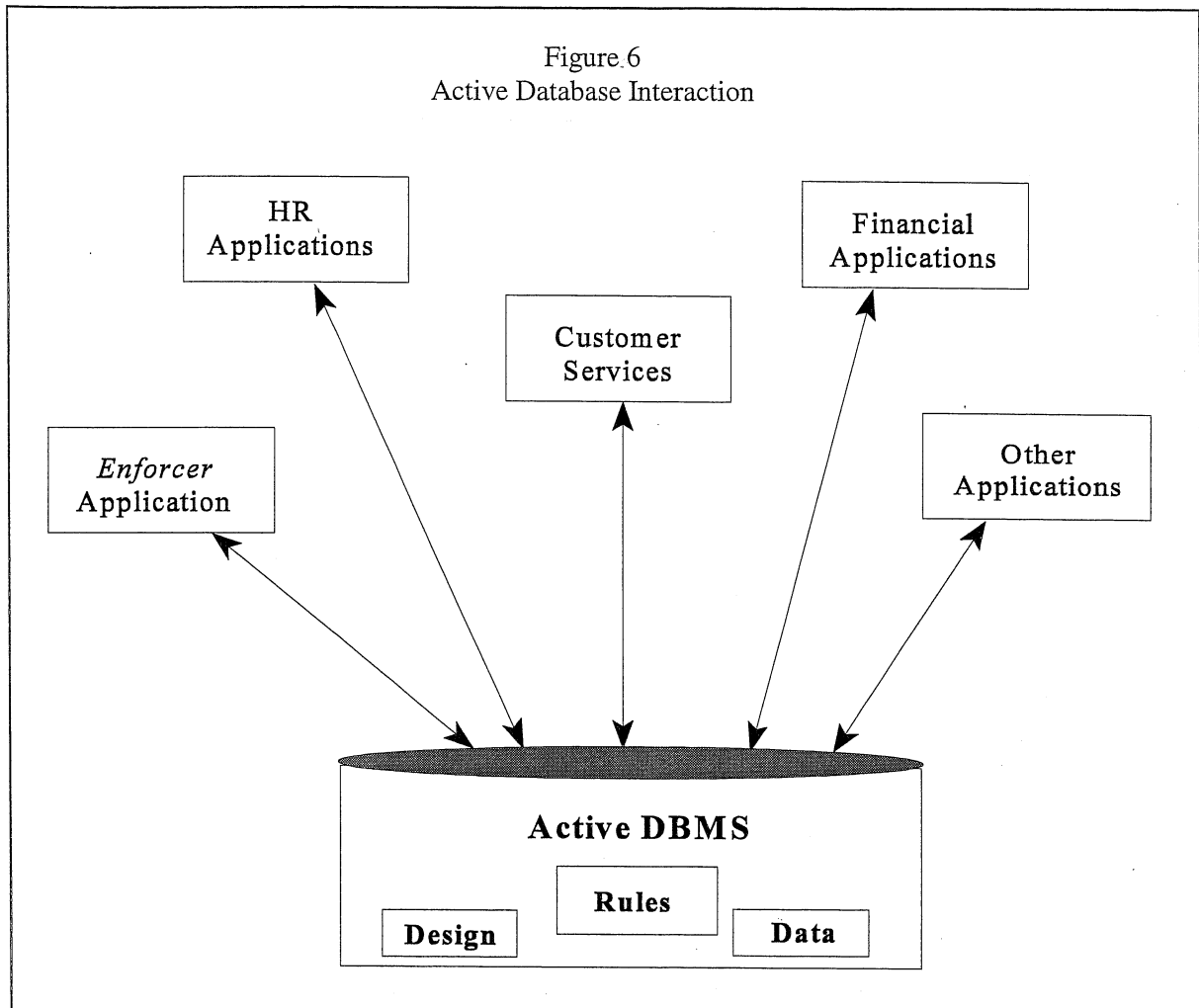
In this rule, a task is assigned to an employee. If the task number is 10, then this employee is automatically assigned to task 11. Perhaps these two tasks are different but should be done by the same employee. This active database rule ensures that the same employee is assigned to both tasks.

In the Enforcer, a production rule is activated upon the creation of a new fund or a change to an existing fund. Once a new fund is created or an existing one is modified, e-mail messages are sent out to each person that might work on the task indicating if the fund change requires some work from them. The fact that a

trigger detects additions or updates to funds, independent of a given application being executed, increases reliability. In other words, no application has to be executing at the time of the fund update for the triggering mechanism to execute.

Active database rules have many advantages over conventional databases. First, business policies can be specified as rules and second, automatic execution of the rules is guaranteed. Therefore there is a much higher probability that business policy is being followed. Second, rules are specified only in the database management system (DBMS) as opposed to each individual application. Figures 5 and 6 illustrate the rule placement differences between conventional and active databases. As shown in Figure 5, a conventional database system contains rules





that are developed individually for each specific application that interfaces with the DBMS. In contrast, Figure 6 shows that the rules along with the design and data are a part of the DBMS itself. This leads to a third advantage, which is rule reusability. Once it is established that the specified rules are correct, they can be reused in similar circumstances. This results in tremendous savings in development time and also enhances the integrity of the system. The integrity of the system is enhanced because one rule is developed, proven correct, and then reused in similar applications as opposed to independently developing several similar rules. ECA rules allow designers, such as those who designed the *Enforcer*, to incorporate dynamic features of an expert system into a database system. However, unlike expert systems, which are usually highly tailored to a single application, active databases can be applied to large scale systems by inte-

grating ECA rules into the DBMS data management features.

### How The System Benefited The Company

The system has received considerable management support ever since it was first proposed. Management initially embraced it with open arms and one member of the management team described it as an "awesome" application. The system has had several positive effects on the company and its business processes. Managers and supervisors now have more confidence that activities will be completed accurately and in a timely fashion. Everyone is notified of activities to be performed and all employees have an enhanced opportunity to complete their work on time.

The system gives non-supervisory em-

ployees a sense of independence and a sense that they are adding value to the enterprise. Each employee can independently track work throughout the system and identify both their work in progress and their completed work. Persons who travel can access the system from a remote location via the Internet and complete their tasks as needed. Similarly, employees working at different business locations can access the system, complete the tasks they are assigned, or query the status of any activity. Thus, the system facilitates decentralization and has provided an opportunity to integrate business operations in several different locations. All these advantages have accrued in an environment that includes considerably less paper work and significantly better audit trails. Overall, the company has addressed the problems and related risks that led to the design and implementation of the *Enforcer*, and in the process created a resource that has streamlined work, enhanced quality, reduced cycle times, and increased employee productivity.

### Conclusion


This case demonstrates how information technology can be used to manage internal business processes in a service enterprise and offers several valuable insights to managers and accountants. First, the case demonstrates that a combination of e-mail technology and database technology offers a viable approach for coordinating business processes. These technologies are generally used independently of each other. Unfortunately, such independence does not allow managers to utilize the full potential of the two technologies. Although the case relates to a company's internal business processes, the combination of e-mail and database management systems may also be used in managing external business relationships. One possible use is in managing customer relationships. By incorporating *event-condition-action rules* into the customer database, a particular event or condition could automatically trigger e-mail to customers. For example, a database might include a trigger that sends e-mail to customers whose account has remained inactive for more than 60 days. This trigger may

be built directly into the database and would operate on the data independently of any application programs.

The case also provides practical insight into the value of a goal-oriented review of an information system. In general, an information system should be reviewed continuously to ensure that the goals of the system remain relevant and that the system continues to be technologically viable. In undertaking a goal-oriented review of its information system, a company should: (1.) Start by considering the original goals of the system and the technology used to drive the system; (2.) Examine the changes that have taken place in the organization since the original system was implemented. If significant organizational changes have taken place, it will be necessary to review the goals of the system; (3.) Examine changes that have taken place in the technology since the original system was implemented. If substantial technological changes have taken place, it may be necessary to reexamine the goals of the system as well as the technology used to drive the system. This allows the company to assess whether new goals should be pursued in response to the new technology. New technology may support business processes that were not previously considered viable; and (4.) Reevaluate both the goals of the system and the technology used to drive the system if there have been major changes in the organization and the technology. In general, the old system will have to be redesigned whenever there are major changes in both the organization and the technology used to drive the system.

Another observation from the ASC case has to do with management and employee support for technological innovations. Generally, managers and employees will support an innovation, including a new information system, if it is perceived to provide a net advantage over the old process used by an organization (Rogers and Shoemaker, 1971; Ramamurthy, 1995). Although managers are keenly interested in monetary benefits and will endorse a project that increases net cash flows, employees will endorse

an information technology project only if it is championed by a key individual in the organization and is perceived as providing them with a net advantage relative to the status quo (Ramamurthy, 1995). Managers at ASC perceived the *Enforcer* as an information technology application that helped in minimizing risks associated with the old paper technology originally used by the company. They also perceived the *Enforcer*, particularly the revised version, as a technology that was helping to streamline business processes, improve internal control, and increase employee productivity. Thus, several of the upper-level managers embraced the *Enforcer* and championed it as a valuable business innovation. Employees, on the other hand, appreciated the extent to which the *Enforcer* rationalized their work, increased their effectiveness, and facilitated their work from remote locations.

While factors such as management and employee support, perceived comparative advantage, and a focused goal-oriented review can be linked to the success of ASC's system, one cannot make the generalization that identical factors will contribute toward successful implementation of similar technologies at other enterprises. Thus, additional research is needed to examine the factors associated with successful design and implementation of information systems projects that leverage Internet technologies for the internal business processes of an enterprise. 

---

*We wish to thank Ed Conrad, Associate Professor, University of Akron, for his detailed comments on an earlier draft of this paper.*

### Endnotes

1. The pseudonym ASC, Inc. is used throughout this paper to refer to the company on which this paper is based.

### References

1. Bidgoli H., "A New Productivity Tool for the 90's: Group Support Systems," *Journal*

- of *Systems Management* Vol. 47 No. 4, July/August 1996, 56-62.
2. Brownston, L., Farrell, R., Kant, E., Martin, N., *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming*. Addison-Wesley, Reading, MA, 1985.
3. Dayal, U., Hanson, E., and Widom, J., *Active Database Systems*, in *Modern Database Systems*, ed. Won Kim, Addison Wesley Publishing Co., ACM Press, New York, NY, 1995, pp. 434-456
4. Klein, H. J., "Two Investigations of the Relationships among Group Goals, Goal Commitment, Cohesion, and Performance," *Organizational Behavior & Human Decision Processes* Vol. 61 No. 1, Jan 1995, 44-53
5. Ramamurthy, K. "Determinants and Outcomes of Electronic Data Interchange Diffusion," *IEEE Transactions on Engineering Management* Vol. 42 No. 4, Nov. 1995, pp. 332-351.
6. Rob, P. and Coronel, C. *Database Systems: Design, Implementation and Management* Course Technology, Cambridge, MA, 1997.
7. Rogers, E. and Shoemaker, F. F. *Communications of Innovations*, New York, NY: Free Press, 1971.
8. Sippl, R., "ODBC and UNIX: Port No More," *Unix Review* Vol. 13 No. 9, August 1995 pp. 28-36.
9. Vinokur-Kaplan, D., "Treatment Teams That Work (And Those That Don't) : An Application of Hackman's Group Effectiveness Model to Interdisciplinary Teams in Psychiatric Hospitals, *Journal of Applied Behavioral Science* Vol. 31 No. 3, Sep 1995, pp. 303-327
10. Wheeler, R. "The Systems Scene: E-Mail Case Studies," *Insurance Brokers Monthly & Insurance Advisor* Vol. 45 No. 5, May 1995, pp. 170-173.
11. Widom, J., Ceri, B., "Introduction to Active Database Systems," in J. Widom and B. Ceri (eds.), *Active Database Systems: Triggers and Rules For Advanced Data-*

- base Processing*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1996.
12. Zajas, J., "A Group Process Assessment for Interpersonal Growth, Communications and Managerial Development," *International Journal of Management* Vol. 11 No. 3, Sep 1994, pp. 773-777.

