By R. G. Eaton

I-CASE BEST SOLUTION FOR 401K TEAM

QUICK AND EFFECTIVE DELIVERY IMPRESSED TWENTIETH CENTURY USERS

The solution was not exactly gift-wrapped, but our investment management company was able to respond to a competitive market effectively with the help of an integrated CASE tool. By effectively, I mean the solution satisfied the user's requirements at the time it was implemented.

This is atypical of what usually happens when, because of a protracted development period, the end product is obsolete by the time it is implemented. The quick response was made possible by a shift to CASE from the traditional tools: from IBM, the CICS transaction monitor, DB2 relational DBMS, and the APPC communications protocol, which allows program-to-program communication over the network; and 3270 "frontware" development tools from Easel Corp., Burlington, Mass., which allow 3270 terminal screens to be more GUI-like in their appearance. While these tools would easily support the requirements, it was determined they could not provide the necessary development speed.

The requirement was the implementation of an audio response unit (ARU) for Twentieth Century Services Inc. of Kansas City, Mo. As well as servicing mutual fund shareholder and managing investments, with applications developed by a portfolio management group, Twentieth Century provides record keeping services for 401K plans. While the mutual fund shareholders have enjoyed the service of an ARU for several years, the technology is new to the 401K record-keeping side of the house. The 401K ARU provides customers with a telephone interface to the computer system. Using a touch-tone phone, customers can enter their account identification and listen to a voice speak their account information, enabling them to gain access to their records without the intervention of a customer service representative.

Because of legacy system constraints, sharing a single ARU between the two business areas of pension management and 401K was not practical. The new application was required to support both an ARU and a customer information system. This customer information system was necessary for the company's customer service representatives who are responsible for answering the questions of 401K participants without touch-tone phones as well as people who would just rather talk to a human than a machine. And yes, the customer information system was required to have a graphical user interface (GUI). Development was also expected to be as quick as if it were instant coffee brewing in a microwave oven; in other words, right now!

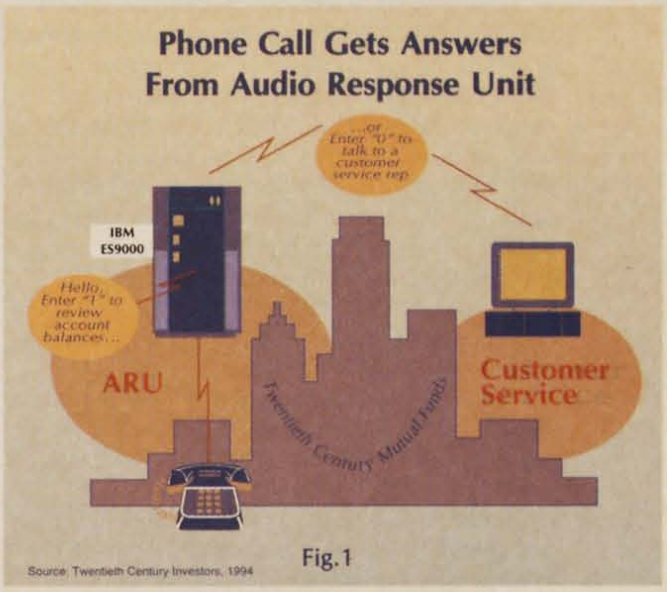


Fig. 1

The audio response unit (ARU) provides customers with a telephone interface to the computer system. Using a touch-tone phone, customers can enter their account identification and listen to a "voice" state their account information, enabling them to gain access to their records without the intervention of a customer service representative.

Fortunately, the business department was equipped with PS/2 workstations. Even though the customer information data resided on a high-end IBM ES/9000 mainframe and the graphic support

R. G. Eaton is an IS architecture and research analyst with Twentieth Century Services Inc., Kansas City, Mo.

was on the workstation, combining the two was technically possible.

AN ALTERNATIVE TO "SPOT" SOLUTIONS

The 401K development team's experience was concentrated in CICS and Easel, so they considered using these "spot" solution tools. As an IS architecture and research analyst within Twentieth Century, my role is to advise the development teams on the tools most appropriate based on our architecture directions. I consider the CICS and Easel tools in use at the time to be spot solutions because they support a single aspect of the development life cycle. The 401K team had no experience with the company's integrated-CASE tool, Texas Instruments' IEF, which had been used by the pension management developers since being acquired in August 1991. That purchase came after a thorough evaluation.

The spot solution tool approach would have involved analyzing the requirements, then coding CICS and Easel programs, and then writing some APPC interfaces to access the mainframe data. Initially, the team considered using an analysis tool along with Easel. The analysis tool that they considered was the IEF.

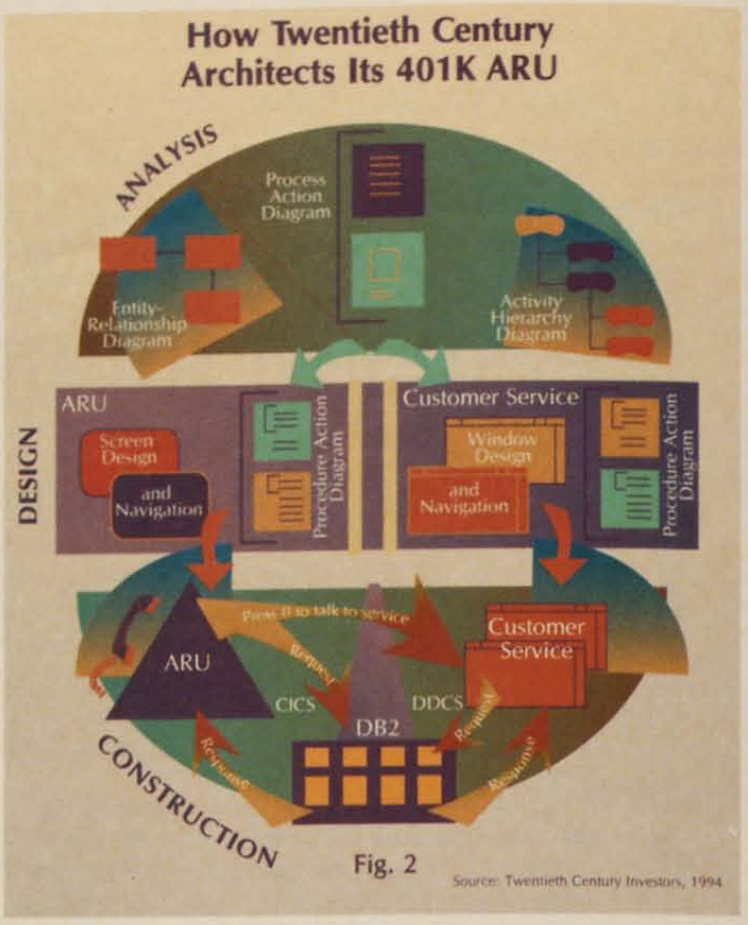
Using separate tools for the analysis, design and construction presents some obvious weaknesses. When the specifications are separated from the actual code, deviations develop. In such an instance, it becomes impossible rely on the specifications for documentation. Instead, the actual code must be used for system documentation. For developers, reading code is not fun, but at least they can muddle through it. Communicating specifications to a user requires something better than computer code. Users need to see their specifications in a entity relationship diagram, an activity hierarchy diagram or flow chart — anything but code.

When the team explained that it wanted to use the IEF to model the data and then code the programs in Easel and CICS, it became apparent that it needed an update of the capabilities of the IEF. To re-cap the requirements, a GUI customer information system was needed to compliment an ARU system. Both systems are logically the same but have different execution platforms.

ONE SYSTEM, TWO PLATFORMS

The code requirements for the ARU included a 3270 data stream. Because the IEF can generate both the GUI panels and the 3270 data stream from the same activity model (created in the analysis component of the tool), the team would not need to write an additional CICS program for the ARU. They could re-use the same "programs" but target them for each different platform. It was obvious that this application was perfectly suited for IEF and the company's first client/server application.

The team's initial approach was slightly modified to use the IEF for complete life-cycle development. The tool can store the analysis results while



Using the IEF integrated CASE toolset from Texas Instruments, developers at Twentieth Century move from analysis, to design and construction, within the same toolset. The IEF tool is also being used to generate code to support communications between clients and servers via IBM's DDCS and DRDA protocols.

maintaining integrity with the actual code. Additional benefits were derived from the code generation features. The development approach became very clear. Using an integrated full-life-cycle tool for the project would be better than a mix-and-match set of spot solution tools.

Using an integrated CASE tool would also eliminate some of the problems of using spot-solution tools. While the IEF will generate all the communication software between the workstation and mainframe today, that was not an option in this project. The communication component had to be hand-coded with APPC. At this point, the team was convinced that code generation from the analysis was a great feat. However, it wasn't thrilled about coding and maintaining the communication module.

Coincidentally, the Twentieth Century technical service people were experimenting with the Distributed Relational Database Architecture, DRDA, features of DB2. A feature of DRDA is Distributed Database Connection Services (DDCS), which provides communication between OS/2 Database Manager (DBM), and DB2. Specifically, it removes physical data address considerations from an application. With DDCS the application accesses a data table without regard to whether the table is in DB2

or DBM. At runtime, the application will find the data. The communication is managed by the DBMS rather than a communication program (such as an APPC program).

MOVING ON TO CASE

After learning of this technology, the team decided to use DDCS to provide the communication between the workstation and mainframe. This required that all application procedures be totally executed on the workstation, leaving the mainframe for data access only. This would enable the 401K development team to use the integrated CASE tool to generate all of the code. Knowing the risks of new technologies, management endorsed the approach. Development of the company's first client/server application to be produced with the help of a CASE tool was launched. Because of aggressive timelines and a team with no CASE experience, the first objective was to select a consultant. There was no time for the team to attend formal training sessions, so it would be necessary for the consultant to transfer skills as the project progressed.

Selecting the right consultant took three to four weeks. The effort was critical to the success of the project and the team's long-term ability to expand and support the application. As expected, there were countless client/server wannabes but few experienced consultants with the people skills to lead a band of CASE virgins.

Because traditional development times can require two years, the requirements frequently change before the application can be delivered. This project lasted only ten months.

The project began with conventional data modeling. All data was maintained by an existing system. The data model represented new decision support (customer information) DB2 tables. Populating and refreshing the tables was accomplished by conventionally coding Cobol batch programs. Because the system was focused on customer information retrieval, activity modeling was limited to the minimum effort necessary to generate code.

Working through countless technical hurdles, the team delivered a client/server systems that featured an effective graphic interface. Because traditional development times can require two years, the requirements frequently change before the application can be delivered. This project lasted only ten months. The users actually got the application that they had originally requested. They were ecstatic that the application was developed so quickly. Compounding their glee, the system included several features beyond the original scope.

Once the customer information system was implemented, the team began the effort to generate the programs for the ARU. According to developer Stephenie Mattly, "We were surprised at the number of objects that we were able to re-use."

EXPANDING THE APPLICATION

As the application expands and changes in response to business, the real efficiency gains should begin to prove the true value of CASE. Training new developers is another expected productivity gain. Because the application requirements are linked to the code, people can understand the application by reviewing the supporting documentation instead of the code.

The team has already learned about the advantages of maintaining an application with an integrated CASE tool. They obtain quick and complete impact analysis from the central encyclopedia. The construction tool generates 100% of the application code and has never had a bug. Because the tool is integrated, the documentation is always correct and reflects the current state of the application. A major concern with maintenance, however, is focused on referential triggers.

The tool will generate all the code to maintain data referential integrity. This is a wonderful feature for developers, but can be cumbersome for maintenance people. When a change is made to the model, the tool generates the entire application. Total generation is necessary to generate the referential triggers. Not all programs may be affected by the maintenance, yet all programs are generated. It is not desirable to move the generated object code for the entire application into production load libraries. The team examines timestamps to determine the changed object code. They only move the actual changed object code to the production load libraries.

A disadvantage of using DDCS is network limitation. Only applications with light usage are appropriate for this type of technology. However, because this application is only an alternative for the ARU, traffic is light. If and when traffic increases, it will be necessary to migrate the application to a different communication platform.

The IEF offers the capability to generate all of the communication code to enable client/server applications. With that feature, the developer selects the program modules that will be clients and the ones that will be servers. Once the modules are packaged, the developer simply generates the program. IEF handles all the communication between the clients and servers.

The project began before the company installed the new IEF client/server generator. When the company is ready for the new code generator, the application should need only minor changes and a code generation.

This application was a suitable introduction to CASE client/server. The transaction volumes were

appropriate and required minimal data maintenance. Essentially, the application is a decision-support system for customers, one whose simplicity is appreciated by the user after having seen a few "Rube Goldberg" client/server proposals.

INTO THE USERS' HANDS

Managing the objects of a model is a topic of protracted discussion. I won't attempt to cover it here. Rather, I'll briefly describe the approach used to launch this application. This description is included because implementing the application was one of the significant challenges that the team faced.

Each object is migrated from the project model to a staging model. The staging model includes objects from the production model. Once the staging model is populated, the application is generated and the resulting programs tested. When the programs are accepted, the production model is archived and the staging model is renamed to become the production model. The applications are generated from the production objects and the generated programs (object code) are moved to the appropriate load libraries.

All the object migration is performed by selecting the objects from a list of candidate objects. Each object may be part of other objects, for example, and each attribute is an object of an entity type. Therefore, when you select an object that is dependent upon another object, it is necessary to select both objects. Because this object selection process requires careful analysis to ensure that all correct objects are selected, the potential for human error is high. For this reason, and others, the IEF toolset provides a trial migration feature. Implementation is a cooperative effort between a development support team and the application developers.

The development cycle was only the beginning for this application. When a feature is added to the application or a business change causes a modification, the team must deliver the new programs via the "sneaker network." The development team is working on that problem by analyzing the possibility of executing the programs from a file server instead of individual workstation and using LAN distribution techniques.

The users are happy, the business is served and the developers have a manageable application. "We now have a framework for future applications," said Kevin Gravino, programming manager. "This exercise provided us with the confidence in the technology," he continued. Client/server applications can be quite manageable when they come in a CASE. ■