SPE
Society of Petroleum Engineers

SPE 027560

Migrating Legacy Petroleum Engineering Applications to Open Systems Environments, or

What Happens When the Petroleum Engineering Software Development Staff Can't Spell C or X?

Todd Little, David Chien, Rory Corbell, M. Ahsan Rahi, Craig Sinclair

Abstract

A case study is presented involving the software life cycle of a suite of commercial petroleum engineering reservoir simulation legacy applications. As with typical petroleum legacy applications, these were designed for DOS or VMS environments, written in FORTRAN, based on GKS for graphics and have hierarchical menu trees.

The hierarchical user interface was replaced with a C++ toolkit to provide a Motif look and feel, while keeping the remainder of the application in FORTRAN. As a result, the application developers that are well versed in FORTRAN can continue to augment the engineering portion of the application, allowing training in C, C++, X, and Motif to be spread out over time.

This migration path has had several major benefits over a complete rewrite of the software. We have been able to produce an open systems application significantly faster than what would be required for a software rewrite. Additionally, this path has preserved compatibility with the legacy application and therefore provided enhanced reliability.

References and illustrations at end of paper

Introduction

Excellent legacy petroleum engineering applications have been developed over time. Although these applications provide good technical value, they often do not run on an open systems platform. When they do, these applications typically retain their original user interface giving them an antiquated look and rendering them awkward to use in today's event driven environment.

Typical petroleum legacy applications were designed for DOS or VMS operating systems, written in the FORTRAN programming language, used the GKS graphics API for drawing graphics into a single window, and based the menuing system on hierarchical menu trees. In contrast, an application designed for open systems typically runs on UNIX platforms, is written in C, C++ and/or FORTRAN, draws graphics to the X window system, and uses Motif for an event driven user interface.

Several options exist for migrating legacy applications to an open systems environment. A case study is presented involving the software life cycle of a suite of commercial petroleum engineering reservoir simulation legacy applications (DeskTop VIP) and the process by which these applications have been migrated to an open systems architecture. The software development started in 1984 as an IBM-PC application, and subsequently over 30 man years of effort have been put into the software development process. The application has been distributed worldwide and has been installed on a wide variety of hardware platforms.

Although the project started as a PC application, an initial design consideration was to be platform independent. A layered toolkit approach fostered a high degree of software reuse and new applications were quickly developed. This design allowed a stepwise migration of the applications to an open systems architecture. It was possible to replace the hierarchical user interface with a C++ toolkit layered on Motif to provide pulldown menus and a Motif look and feel, while keeping the remainder of the application in FORTRAN. The C++ toolkit was designed so that very little C++ or Motif knowledge would be required to modify the user interface. The application developers that were well versed in FORTRAN could continue to augment the engineering portion of the application, allowing training in C, C++, X, and Motif to be spread out over time.

This migration path has had several major benefits over a complete rewrite of the software. Since only the user interface has changed, the majority of the application remains unchanged maintaining the compatibility with previous versions and the integrity of the application. Consequently, it has been possible to produce an open systems application significantly faster than what would be required for a software rewrite while preserving compatibility and reliability.

Description of the Legacy Applications

In 1984 J.S. Nolen and Associates began a development project to provide a suite of graphical pre- and postprocessing software to be used with the VIP reservoir simulation software package. The two applications that were the initial focus of the development effort were oriented towards providing tools for the reservoir engineer to assist with the time consuming aspects of the reservoir simulation process and to provide graphical insight into the reservoir simulation.

The graphical post processor, SIMOUT, allowed for viewing XY plots for well production along with map views of simulator grid block properties. With this visualization capability, the reservoir engineer could quickly analyze the results of a reservoir simulation study greatly expediting the simulation process. By using the visual tool, the engineer was also able to explore the simulation study to gain new insight into the physical behavior of the reservoir.

Another time consuming part of the reservoir simulation process, and also the step most likely to induce error, is the generation of the reservoir simulation gridblock description. To provide an engineering tool for this facility, the pre-processing package GRIDGENR was developed. This application has been used effectively for many reservoir simulation studies and has been shown to have saved a great deal of effort.[1,2]

Consistent with the original plan, it became obvious that additional pre- and postprocessing applications would be beneficial for the reservoir engineer. Applications that have been subsequently developed include a graphical relative permeability curve editor, a PVT property editor, and a utility for entering general simulation setup parameters.

Initial Application Architecture

At the beginning of the development process, the advent and popularity of the IBM-PC dictated that MSDOS platforms would need to be supported. It was also realized that it would be very limiting and shortsighted to focus development strictly for the PC. In fact, it was clear from the outset that it would be desirable to provide the same tools on a VAX/VMS platform. This situation provided a bit of a dilemma since at the time there was no clear path that would provide portability between these two platforms.

When the development was started, the staff had experience only with the FORTRAN programming language. One hope for portability was the Graphics Kernel System (GKS). GKS is an ANSI standard graphics drawing Application Program Interface

(API) with defined bindings for FORTRAN. However, at the time there was no GKS available for MSDOS. There were indications that GKS would eventually be available for the PC and thus development continued along that line. It was quickly realized that the full featured GKS environment used too much memory to allow the building of commercial reservoir engineering applications. The 640K memory limit meant that overhead needed to be eliminated whenever possible. After further application design, it was determined that the GKS model was acceptable, but that the retained segment capabilities offered by the full Level 2 GKS were unnecessary. As a result, a GKS-like toolkit was developed to provide the specific graphic functions required by the engineering applications. This toolkit was designed to be able to use a Level 0 GKS implementation, and also allowed for the development of Level 0 GKS device drivers as needed.

The experience with the graphical toolkit led to the development of low level toolkits to handle the machine dependent operations such as file handling and screen operations. These low level toolkits provided the basis from which to generate higher level toolkits such as a textual forms tool, a graphical menu tool, an XY plotting tool, a contouring and curve fitting tool, and a database access tool. This toolkit layering is shown in the Application Architecture Diagram in Figure 1.

Architectural Diagram of Application Suite

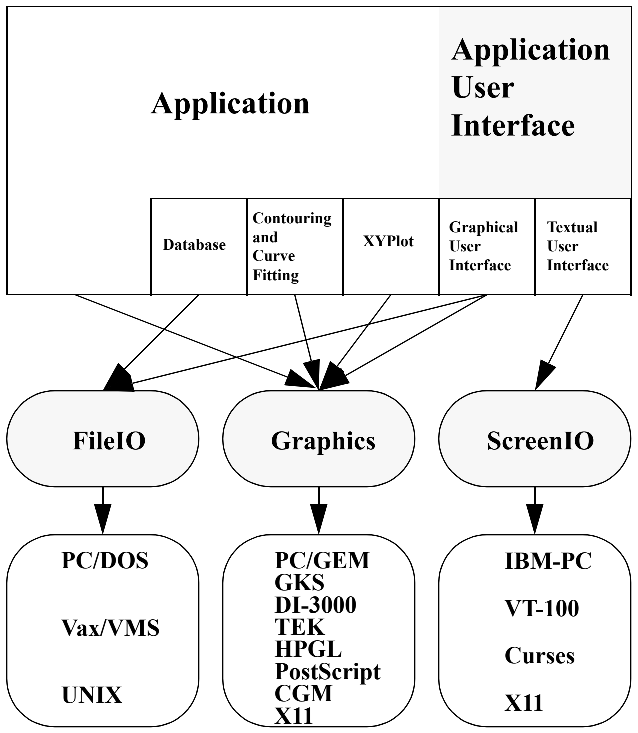


Figure 1

Previous Migration Paths

The layered toolkit approach improved application portability and encouraged software reuse. Application developers found that they could build their applications much more rapidly by utilizing the toolkit components. Consequently, all of the pre- and postprocessing applications were built upon the toolkits. This layering also provided a mechanism to insulate the application from nearly all machine dependencies. Since only the toolkits needed to be migrated, the effort required to port the applications from one operating system to another, or from one graphics device to another, was significantly reduced

The original software development effort was predominantly on MSDOS PCs, with the graphics development using the OMNIcomp graphics controller. This provided adequate graphics display, but hardcopy output was only available via screen dump. To provide for scalable hardcopy and additional graphics displays, the required component of the GKS-like layer were developed utilizing the GEM software drivers. Once the GKS-like layer was developed, all the applications were able to have this new hardcopy capability.

As interest in the applications grew, so did the need to support additional platforms. The other major platform at this time was VAX/VMS, and the introduction of the MicroVAX furthered its popularity. For these applications, the OMNIcomp graphics controller was also used on the MicroVAX, and hardcopy was provided by developing the graphics drivers for the HP plotters and for Tektronix printers, or by utilizing a full GKS implementation for both the graphics display and the hardcopy output.

The screen I/O interface provided another challenge for portability between MSDOS and VMS environments. To provide this facility, the screen I/O toolkit was built to support the VT100 and the applications were not altered. Similarly, file I/O operations which were optimized for the PC were built using generic FORTRAN for the VAX.

During this phase of the software life cycle, the following platforms and graphical systems were supported.

Platforms:	PC/MSDOS, VAX/VMS, Apollo/Aegis, IBM-RT/UNIX, IBM/VM, Clipper/UNIX, HP3000/UNIX
Graphics:	OMNI(PC,VAX), GEM(PC), GSS(PC), GKS(VAX), DI-3000 (VAX,IBM/VM), Tek(VAX), CIT(VAX), VT340(VAX), Lexidata (VAX), Starbase(HP), GPR(Apollo), GPX(VAX), Calcomp (VAX,UNIX), Versatec(VAX,UNIX), HPGL, PostScript, CGM, X11(UNIX,VMS)

Open Systems Environment

The ability to migrate the toolkit has enabled each of the applications to be usable in an environment in which it was not initially expected to be used. In fact, simply by porting the underlying toolkit it was possible to allow the applications to operate in an open systems environment of UNIX and X11.

The basic functionality of the applications in this environment was considered valuable to the engineers. However, it was clear that the graphical user interface was difficult to navigate and incompatible with the Motif interfaces to which users were becoming accustomed. The fundamental layering of the menuing toolkit did not allow a trivial port to a Motif user interface. It would have been possible to do a direct translation of the menu toolkit to Motif, but that would have resulted in the exact same menus but with Motif buttons. This cosmetic enhancement was considered to be of minimal value.

Alternatives Considered

Two basic alternatives were considered to reach the desired objective of a Motif user interface. One option was to completely rewrite the applications and design them to use Motif from the outset. Such a rewrite would be done using either the C or C++ programming languages, a much more natural fit for open systems environments and Motif than FORTRAN. The other approach involved rewriting just the user interface component of the applications, leaving most of the remainder of the applications untouched.

Before deciding which approach should be taken, it was necessary to inventory the resources available for the conversion and the level of experience. At the time (1992) the decision to migrate to Motif was made, there were 7 developers dedicated to the overall project. As can be seen in Table 1, of the 7 developers, all had working knowledge of FORTRAN, and 5 had extensive background with FORTRAN. Most of the developers had some exposure to the C language, yet few would have been considered proficient. Even fewer were proficient in C++ or Motif.

Table 1:

Knowledge	Some	Good	Proficient
FORTRAN	7	6	5
C	5	3	2
C++	3	2	1
Xlib	3	3	1
Motif	2	2	1

What do you do when your engineering development staff cannot spell C or X?

Figure 2 depicts a set diagram of the collection of expertise with regard to the applications, C, C++, and Motif. Not surprisingly, those most proficient in C++ and Motif were the least proficient in FORTRAN and also the least familiar with the legacy applications. The converse was also true in that those most familiar with the applications generally had very little background with C, C++, or Motif. As can be seen, the intersection of those that had expertise with the applications, C++, and Motif was very small subset of the overall collection of developers.

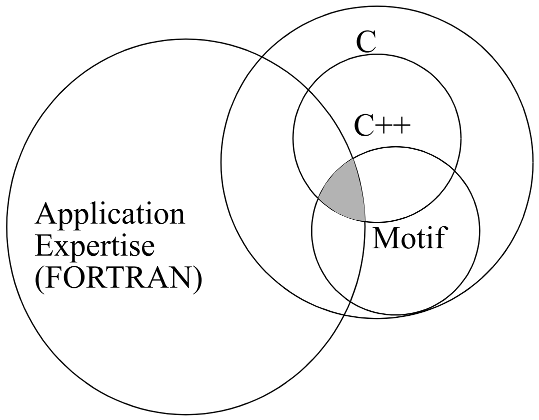


Figure 2

Migration Strategy

Given the limited Motif expertise available to migrate the applications to Motif, it was necessary either to train the application developers in C++ and Motif, or to provide them with tools which allowed them to develop the needed enhancements. Based on previous experience, it was estimated that it would take 6 months for a C developer to become proficient in C++. The estimate was even longer for a FORTRAN programmer with limited exposure to C. A similar effort would be necessary to become proficient in Motif.

At the time the migration strategy was being developed, the developers who were proficient in C++ and Motif were working on a 3D visualization component to the pre- and postprocessor. As this was an entirely new product, it was designed in an object oriented fashion for an open systems environment. The development included the creation of an object oriented user interface where the objects were comprised of Motif widgets. The interface toolkit is not simply a wrapper around Motif; it is a framework whose structure and functionality is designed from the application's point of view. This "application-oriented"[3] approach hides the complexities of X and Motif and provides a much more natural fit to the application. For example, the toolkit allows the simple one step creation of a fully customized Motif main window. The application developer defines the interface using a declarative form while retaining the ability to dynamically alter components of the interface. This object interface greatly expedited the process of developing the user interface component of the 3D application by providing reusable objects. In addition, the object layering kept X11 and Motif related issues isolated. Thus a foundation was laid for an application framework upon which current and future developments could be based while still maintaining an insulation of the applications from changes in user interface technology.

The success with this interface toolkit indicated that the same toolkit would be usable within our legacy applications. Since the object interface insulates the application developer from Motif, the learning curve for the user interface component could be greatly reduced. By focusing development on a reusable toolkit, it was possible to make optimal use of the limited C++ and Motif expertise available.

Through the use of the toolkit, it was determined that it would be possible to keep a majority of application framework intact. This was particularly true in those instances where the application user interface was isolated from the application core. Interaction between the core and the user interface was through a small set of functions defined in the user interface objects. Therefore, the application developers familiar with the application core could continue to enhance the application using FORTRAN and not be required to be proficient in C++ or Motif.

Case 1: GRIDGENR

The first legacy application migrated in this manner was GRIDGENR, the reservoir simulation grid generator, editor, and property calculator. Over 20 man years of development had been invested in this application and the underlying software tools resulting in over 100,000 lines of FORTRAN and C code. The application is highly interactive and graphical in nature. Interaction within the graphics area was generally well liked by users; however, the application had grown to contain over 500 menu choices and navigation through the menus was considered a significant drawback. It was therefore an excellent candidate for restructuring using Motif menus.

After a careful study of the original GRIDGENR code, the menu items and the corresponding action routines were identified and extracted. For the most part, the user interface component was already isolated from the application core or action routines. A revised menuing system appropriate for an open systems application was then designed.

GRIDGENR Restructured Architecture

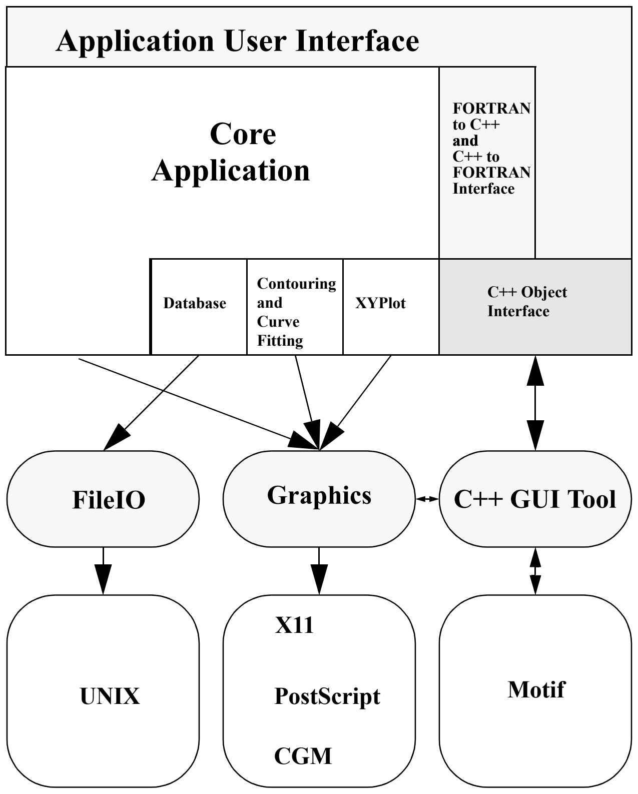


Figure 3

The overall restructured application architecture is depicted in Figure 3. As can be seen by comparison with the existing architecture (Figure 1), the core of the application remains intact. The user interface component, however, is completely restructured and rewritten. The user interface now drives the application action routines, whereas previously the application drove the user interface. The application user interface utilizes the C++ user interface tool, and communicates to the action routines and vice-versa via a C++ to FORTRAN interface layer.

The complexity of the menu system forced a further enrichment of the toolkit features. Thus the toolkit evolved and matured as needed, side by side with the application. As a side effect, through continuous but limited exposure to the Motif/C++ toolkit, the application developers progressed on a comfortable learning curve, advancing to a point where they directly began making enhancements to the toolkit as needed.

Two developers were involved is this process, one developer familiar with the toolkit but not the application, and one developer with good application experience and some familiarity with C, but very limited exposure to Motif. It took approximately 6 months of elapsed time and 7 man-months of effort to complete the migration and conversion process. The relative efforts required were 2 man-months for the toolkit developer and 5 man-months for the application developer. This total time included the start-up time for the application developer to obtain the required proficiency in Motif, C++, and the existing toolkit.

While these developers concentrated on the user interface conversion, another application developer continued enhancing the FORTRAN core of the application. Since the enhancements were made primarily to the application core and not to the user interface component, very little effort was required to merge these enhancements back into the Motif version. Once the Motif version was stable, the FORTRAN enhancements could be made directly in the new version. Furthermore, since the object interface toolkit simplified the programming of the user interface, very little time was required for the FORTRAN developers to become proficient in applying the new user interface. The application developers could make changes to the user interface without being highly proficient in C++ or Motif.

Case 2: PREXEC

At approximately the same time as the GRIDGENR conversion was begun, it was determined that it would be valuable to have an application (PREXEC) to provide an interactive interface for entering and importing of production history data and other simulator recurrent data. Since there were no software developers available who were already proficient in Motif, it was decided that it was more important to produce this application quickly than it was to have a Motif application from the outset. It was, however, desirable for this new application to eventually be migrated to a Motif interface. Therefore, in the short-term, functionality had to be provided using the existing software toolkits.

The user interface components required for this application were pulldown menus, textual entry panels and selection boxes. These tools were all available within the existing toolkit and had corresponding components within the Motif widget set. The design and original development using the existing tools resulted in an application written in modular FORTRAN but displaying many of the look and feel qualities of a Motif based application. By this stage in the development process, the PREXEC application consisted of about 30,000 lines of FORTRAN code with 25 menus, 120 menu choices, and 17 data entry panels.

The application was designed with the intent of minimizing the code changes to the application when it was migrated to Motif. As can be seen from the architecture diagram in Figure 4, there is very little difference in the resulting architecture when compared to the original architecture (Figure 1).

This migration, while in many ways quite similar to the one chosen for GRIDGENR, has one significant difference. Whereas the GRIDGENR application was restructured to remove the user interface from the application core, the route chosen for PREXEC kept the application and the FORTRAN interface to the tools intact and instead replaced the underlying tools.

While this structure significantly simplifies the conversion process, it does have the disadvantage of restricting the application. Since the toolkit is designed as a subroutine task and is not designed to have application callbacks, each of the toolkit components operates in a modal manner where one operation must be completed before the next operation can be chosen. For this application, it was decided that the restriction of modality was not a significant drawback, and may indeed be an advantage in that it helps guide users through the application.

PREXEC Architecture Diagram

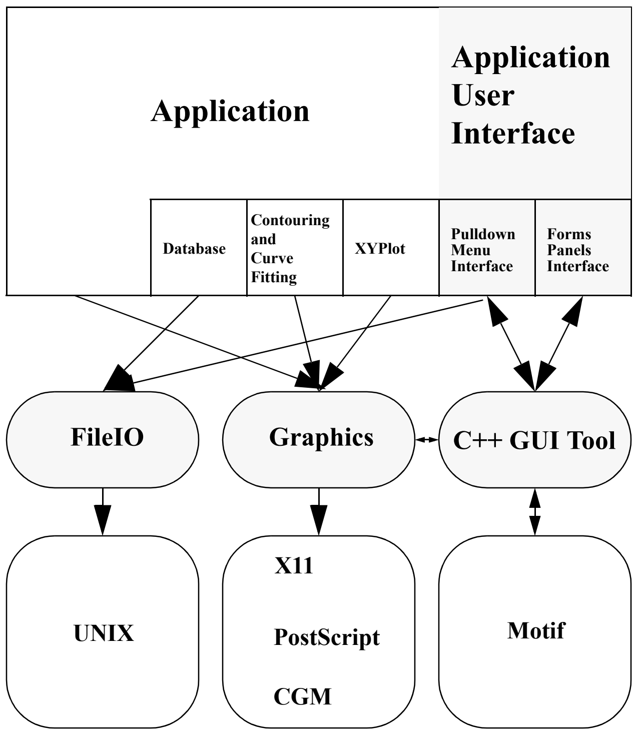


Figure 4

Meanwhile, as the application was being developed with the existing menuing toolkit, the toolkit was simultaneously being migrated to Motif by a developer proficient in Motif. This toolkit conversion was layered on the user interface objects utilized for GRIDGENR. Each function of the existing toolkit was replaced with a corresponding Motif widget.

Once the toolkit conversion was finished, the final modifications to the PREXEC application were also made by the toolkit developer. Because of requirements of the C++ language, the main routine became a C++ stub which called the PREXEC FORTRAN application. This restructuring and a few other minor modifications were the only necessary changes to the application. However, once the conversion was done, it became obvious that some additional application modifications would be desirable in order for it to behave more like a true Motif application.

The conversion of the toolkit required the full time effort of the toolkit developer for 4 months. An additional 1 man-month of effort was spent to polish the resulting application, with that activity being shared between the toolkit developer and the application developer. Subsequently, several other applications which were based upon the existing toolkit have been converted with each application taking approximately one man-week to convert.

Case 3: HVWELL

As the development of PREXEC reached completion, a client inquired if this technology would be applicable to one of their internally developed engineering applications. Their application was a FORTRAN program for assisting with the reservoir simulation data entry requirements for a single well simulation model of a vertical or horizontal well[4,5].

Development had been in the IBM-VM environment using SPF panels as the menu and data entry mechanism. It therefore had not been developed using the described toolkits. The application consisted of over 3,000 lines of FORTRAN code and 20 data entry panels. Several man-years of effort had been invested in this technology and the users had been very satisfied with the application. However, because of the restriction to IBM-VM and SPF panels, many users that would have liked to have had access to the application on an open systems platform discovered that it was not available.

The structure of the application indicated that the conversion process utilized for PREXEC could be applicable for this application as well. Since the toolkit had already been migrated to Motif, the majority of the conversion effort was to replace the SPF data handling facilities that were no longer available by utilizing the toolkits. The calculational portions of the application were changed only where absolutely necessary to implement the interface. This reduced the amount of debugging required on the client's already proven application code.

Through this process, a Motif application was generated using 1 man-month of effort. Furthermore, continued development of the application will be possible by the client using known FORTRAN techniques.

Conclusion

Several legacy petroleum engineering software applications have been successfully migrated to an open systems environment using these techniques. A major restructuring of the applications for open systems would not only have been a substantial undertaking, but would also have required a significant training investment and corresponding time delay for the developers to reach sufficient proficiency. By focusing the limited resources that had the necessary Motif proficiency on the toolkit restructuring, the overall migration task was greatly expedited.

A second major advantage is in the area of application compatibility and integrity. Since the majority of the modifications are to the user interface component, the rest of the application remains unchanged. The static nature of the application core insures compatibility with previous versions and maintains the integrity of the application.

One additional advantage is development continuity. Those developers already familiar with the application can continue to enhance the application using technology with which they are proficient. This enables training and experience with new technologies to be spread out over time and integrated with their existing knowledge base. Likewise, it removes the requirement that all developers be proficient in all technologies.

A potential advantage of the object interface that has begun to show recent dividends is the insulation of the application interface from Motif. Motif itself is not a stagnant system and the changes from one release to another require changes only at the toolkit level. This greatly reduces the effort required to keep pace with changes in user interface technology, and could potentially allow for migration to other user interface technologies as they become the choice *du jour*.

The main limitation to the approach taken in the PREXEC and HVWELL conversions is the continuing modality of the applications. Although not a problem at this time, future desired enhancements to the applications may not be possible due to this restriction of the current implementation.

In summary, the layered toolkit solution has allowed for the migration of several legacy applications to an open systems environment. Previous migration experience shaped the original design of the legacy applications and greatly eased the migration path by localizing the required modifications. The design of the underlying user interface toolkit was also flexible enough to relatively easily migrate an application which was not designed originally to use the toolkit. This process has retained the valuable components of the legacy applications. In this manner, open systems applications have been produced significantly faster than what would have been required for a software rewrite, while still preserving compatibility and reliability with the valued legacy applications.

Acknowledgments

The authors would like to acknowledge the efforts of all of the developers that have contributed to the development and enhancement of the legacy applications. It is because of their efforts that it has been worthwhile to migrate these applications to an open systems environment. We would also like to thank Ben Wang and Chester Hermes with Texaco Inc. for their assistance with the HVWELL conversion.

References

1. Choo, Y.K., Wethington, W. B., and Lederer, M. C., "Application of Preprocessing Software in the Initialization of Kuparuk Full-Field Model," paper SPE 22311, presented at the 1991 SPE Petroleum Computer Conference, Dallas, Texas, June 17-20.

2. Hazlett, W. G., Johnson, R. S., Sibley, M. J., Thompson, J. V., Hrkel, E. J., and Bazzari, J. A., "The Integrated Work Team Approach to Performing Reservoir Simulation Studies," paper SPE 26224 presented at the 1993 SPE Petroleum Computer Conference, New Orleans, Louisiana, July 11-14.

3. Foody, M., "Cross Platform portability: Look before you leap," *The X Journal* (November-December 1993) 96.

4. Wang, B., "A Parametric Study of Gas and Water Coning in Vertical and Horizontal Wells," paper presented the 1991 Indonesian Petroleum Association, October 1991.

5 Wang, B., Markitell, B. N., and Huang, W. S., "Case Studies of Horizontal Well Design and Production Forecast," paper SPE 25567 presented at 1993 Middle East Oil Technology Conference, April 1993.

GRIDGENR Before Conversion to Motif

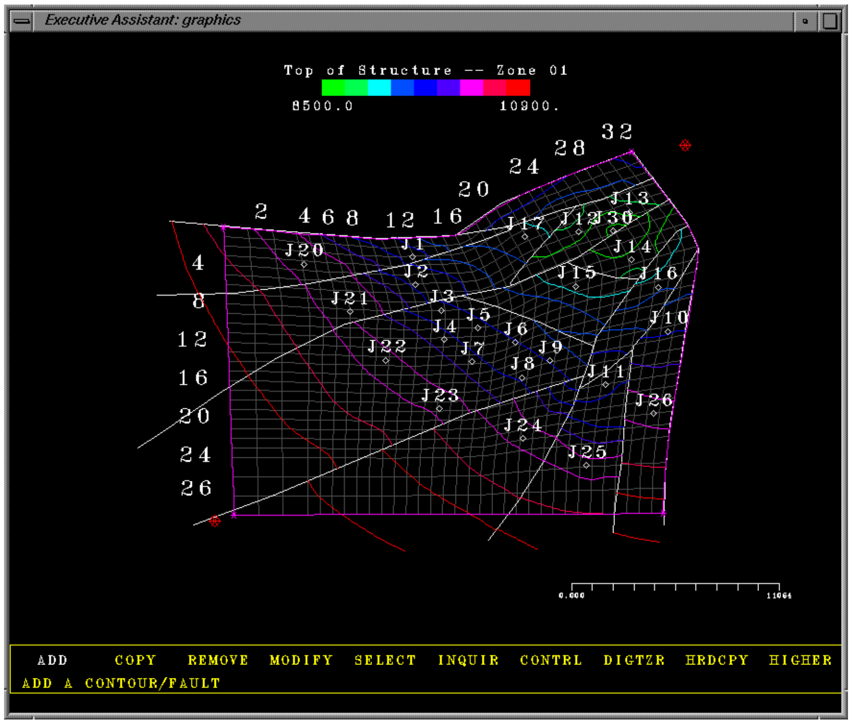
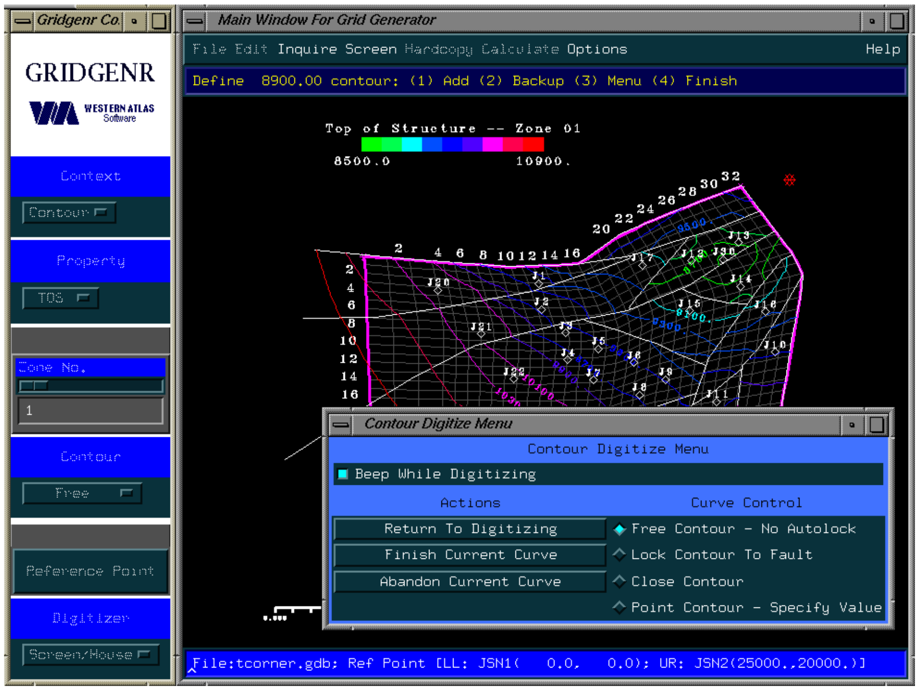

Executive Assistant: graphics
Top of Structure -- Zone 01
8500.0
10900.
ADD COPY REMOVE MODIFY SELECT INQUIR CONTRL DIGTZR HRDCPY HIGHER
ADD A CONTOUR/FAULT

Figure 5

GRIDGENR After Motif Conversion



```
Define  8900.00 contour: (1) Add (2) Backup (3) Menu (4) Finish
```

```
File:tcorner.gdb; Ref Point [LL: JSN1(    0.0,    0.0); UR: JSN2(25000.,20000.)]
```

Figure 6