

1045

Salvaging, Maintaining & Updating Your Legacy Systems

By: Charles Finley
Open-Ended Systems Corporation
546 N. Oak Street • Inglewood, CA 90302
1-310-419-5903 • 1-800-876-5975

<http://www.oesc.com> • cfinley@oesc.com

Introduction

Simply stated, a legacy application is an application of value that has been passed on to the next generation. Many of these applications are old (many were developed in the late 1960s and early 1970s) and are showing signs of age and years of patching and fixing. These systems are vital to the survival of several organizations, however, they are becoming increasingly expensive to maintain and operate (some systems take several months for a simple enhancement such as a new calculation). As the business pressures to provide flexible and timely information for management decisions and operational support grow, the inadequacy of legacy applications is highlighted. In addition, off-the-shelf software and flexible C/S architectures are showing the weaknesses of legacy applications. However, legacy applications are the workhorses for most organizations and cannot be thrown away (it is difficult to throw away applications that support critical services such as billing, inventory control, payables/receivables, and purchasing). This presents a serious dilemma: You cannot live with them or without them.

The year 2000 challenge makes dealing with legacy applications effectively even more important. In many cases organizations have no choice but to perform some amount of reengineering of an application, that which is required to fix the year 2000 date problem.

Other business needs may also make retaining legacy applications desirable or necessary. The most common is the need to rehost an application because of a discontinued computer operating system. Users of Wang or Prime computers have faced this situation.

The paper assumes that the reader is interested in preserving some aspect of a legacy application and has decided to either reengineer and/or rehost the application. The three tasks addressed by this paper are: 1) The desire to either increase the functionality of the application by providing client/server, Web access, converting to another programming language and/or relational switching to relational database access. 2) The need to rehost an application. 3) Conversion of the application and data for year 2000 compliance. The paper examines three methodologies for facilitating these categories of application reengineering:

1. Conventional toolsets for code and data migration
2. Surround technologies
3. Automated code and data migration

The paper then goes on to describe how automated code and data migration are accomplished by using ViaNova and Via2000, two products of Denkart as examples.

Legacy Application Concepts

What is a Legacy Application?

We will use the following definition:

Definition: *A legacy application is an application of value inherited from the past.*

Specifically, legacy applications are a class of applications that are:

- Crucial to the day-to-day operation of corporations
- Heavily invested over the years and cannot be simply "wheeled out to the parking lot"
- Large (e.g., millions of lines of code, thousands of programs)
- Old (e.g., 5 to 20 years old, some are older)
- Used heavily (e.g., thousands of transactions per day)
- Not well documented and difficult to understand (documentation changes have not kept up with code changes)
- Inflexible, costly, time-consuming, and risky to maintain and change
- Based on older database technology (e.g., IMAGE databases) or no database technology at all (e.g., KSAM files)
- Written in older programming languages (e.g., COBOL, FORTRAN, and assembler)
- Based on text-based user interfaces (e.g., VPlus screens) instead of GUI (e.g., windows)
- Vertically integrated and "monolithic" (e.g., tightly coupled user, process, and data management)
- Repositories of years of corporate experience and practices (e.g., many business rules are embedded in the legacy code)

Many business applications that were developed in the 1970s and early 1980s are good examples of legacy applications. Legacy applications are not necessarily very old (during my work assignments, I have heard statements such as "non-Java applications are legacies," "anything not OO is legacy;" and "anything not using the Web are legacies:' In fact, legacy applications will always be with us (some of the new OCSI applications will be legacy in a few years, if they survive). Due to this reason, we will emphasize the key principles in this paper that can be used to reengineer any existing application.

Defining Reengineering

Reengineering means different things to different people. We will use the following definitions of reengineering:

- **Business process reengineering** is concerned with the conversion of business processes, and not of computer processes.
- **Software reengineering** is concerned with conversion of the form and not of the functionality of the software.
- **Data reengineering** is concerned with conversion of data format and not its content.
- **Application reengineering** means that the application functionality does not change but its form changes. Application reengineering can involve data and/or software reengineering.
- **Legacy application reengineering** means a combination of approaches such as access, integration, rearchitecture, and migration strategies.

Legacy Application Reengineering Methods

The reengineering methods needed to deal with legacy applications fall into the following categories:

- **Conventional** toolsets for code and data migration
- **Integrate** Consolidate them into the current and future applications by access in place
- **Automated** code and data migration

The exact mix and sequence of these approaches to form a strategy are based on a combination of business drivers, the technical status of the legacy application being considered, the flexibility and growth requirements, the corporate attitude toward IT reengineering, and several other business as well as technical issues.

Method 1: Conventional Toolsets for Code and Data Migration

Conventional toolsets for code and data migration are perhaps the most commonly used and least efficient. Tools include editors and compilers and not much else. This approach is only useful for small applications. It is the most time consuming and expensive approach. We will not discuss it in this paper.

Method 2: Provide Access in Place (Access/Integration)

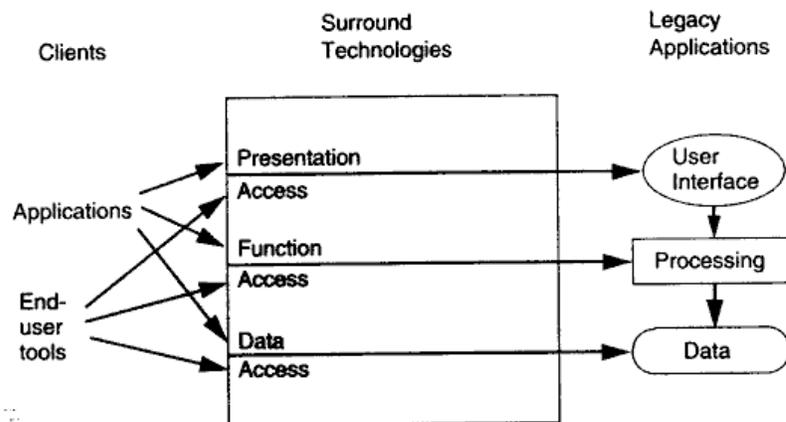
In many cases, access in place is the best approach to deal with legacy applications. This approach allows access to legacy data and integration of legacy applications with new applications and tools without any modification to the legacy applications. This merging old with new strategy is facilitated by surrounding the legacy applications with mediators such as gateways and object wrappers. Access in place is useful for the following situations:

- The business value of the data and/or the application logic is high.

- The access to needed data is urgent and cannot be postponed until after the completion of migration.
- The needed data can be accessed by client applications and end-user tools by employing off-the-shelf technologies.
- The data query requirements are low (most access in place solutions do not work well if a very large number of new queries start arriving at the legacy system).
- The data currency requirements are high, that is, users need to access the most recent copy of data (a shadow database or a data warehouse is not adequate).

Access in place is achieved through "surround" technologies (also known as "mediators") which hide the legacy application details from client applications and end-user tools (see Figure 1). Appropriate surround technologies, such as the World Wide Web (WWW), can considerably ease the task of integrating legacy applications with newer applications (users and programmers do not know whether the data were retrieved from an old IMS-based system or a more modern object-oriented system). It is naturally important to choose the most appropriate surround technologies for access in place. The chosen technologies can employ middleware such as Web, CORBA, and SQL gateways. It is essential to bind the solution approach by setting limits on what features will be included in the surround technology and when these features will be needed. Analysis of the organizational issues (staff estimates, training, roles, and responsibilities) and estimation of time and cost needed for implementation and deployment are also of key importance. We will discuss access in place and various integration approaches, especially through the Web, in the next chapter.

Figure 1 Surround Technologies for Access in Place



Method 3: Automated code and data migration

In some cases, access in place is either not possible or is not the most desirable approach to deal with legacy applications. Automating code and data migration approach allows access to legacy data and integration of legacy applications code even if the code is transformed into another language. In this case the user can choose the extent to which the application design is modified if at all. This merging old with new strategy is facilitated by using sophisticated software to map the programs and data to the new target environment. Automation is useful for the following situations:

- The business value of the data and/or the application logic is high.
- The access to needed data is urgent and cannot be postponed until after the completion of migration.

- The needed data can be accessed by client applications and end-user tools by employing off-the-shelf technologies.

Automation Enabling Technologies - ViaNova Components

ViaNova is a highly specialized software environment that has been created by Denkart to completely automate the conversion of complex source code from one environment to another. Source languages include COBOL, RPG, PL1, Mapper, MVS JCL, Wang JCL and HP VPlus. Target languages include C and other versions of COBOL. Target environments include many variations of UNIX and Windows NT among others. This workbench has been used to migrate applications from one computer, to migrate from

Figure 2 - ViaNova Components

- **ViaNova Components**
 - **STENO**
An expert system used to generate a multitude of Cross-Compilers.
 - **Cross-Compilers**
A set of language translators built with Steno, which allow the conversion of source code into the desired language(s) for the target platform.
 - **Emulators**
A set of tools allowing emulation where full cross-compiling is not desired.
 - **Wingspan**
A Screen Management System which emulates existing user interfaces on the new platform.
 - **Data Conversion Tools**
Steno allows the generation of programs to perform automatic conversion/loading of application data.
 - **Record & Play**
A script generator allowing for automated program testing and analysis of performance characteristics on the new platform.

terminal-based to client/server and, in a modified form (see Via2000) to automate the Year 2000 conversion process. This breakthrough technology dramatically slashes the time required to perform complete migrations. This section describes the ViaNova software environment and in so doing shows the key components necessary to enable automation of application reengineering.

ViaNova, helps its users reengineer applications to state-of-the-art Open Systems Information Technology such as UNIX, Windows NT, RDBMS, client/server and graphical user interfaces (GUI). Through an expert system based methodology, ViaNova provides a flexible rule-based approach for language conversion and systems migration. ViaNova is routinely used by worldwide corporations to rehost and re-engineer legacy applications from IBM, Unisys, Siemens Nixdorf, Wang, and others to newer platforms such as UNIX or Windows NT.

Here's how it works:

STENO

- STENO is an expert system that produces a Cross-Compiler specifically for your target environment. It has four important functions. First, it is a lexical analyzer that probes source code for the underlying structures and processes that constitute the program logic. Second, it is a process converter, transforming processes written in non-standard code to standard code equivalents. Third, it creates a data dictionary so data can be accessed in a new environment. Finally, STENO creates a support library of all the functions and intrinsics your application needs in order to perform in its new environment.
- STENO generates a custom compiler based on the code you want to regenerate. By thoroughly analyzing your existing code, STENO can isolate syntax, standards, and styles which it incorporates into a Cross-Compiler for your applications. This approach to code regeneration provides for a context-sensitive conversion, preserving the rationale behind your programming standards and styles.
- STENO operates on a knowledge base that is fed with three types of information. First, syntax and lexical rules are defined for the original source code. Next, production rules are constructed that provide direction on how to convert elements of the source language into the destination language. And finally, programming standards and company related rules can be described for transport to the new code. Furthermore, if the analysis shows areas that can be improved upon, STENO will generate a report for those aspects of the program benefiting most from optimization or functional enhancements.

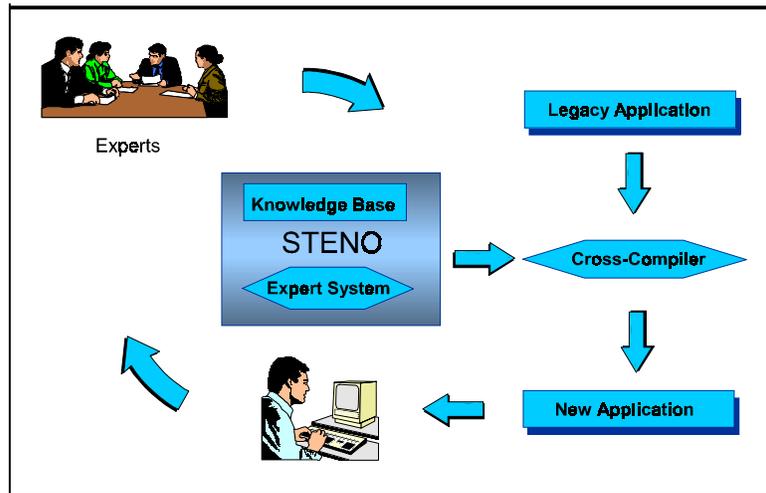
Building the Cross-Compiler is a three-step process.

- **Parsing:**
Source files are entirely parsed into a series of definitions in Backus-Naur Form (BNF) equivalents. STENO performs this function by creating tree structures composed of variable declarations, procedures, statements, and expressions. These imbedded structures continue to be developed until the application source file is completely expressed in a BNF equivalent.
- **Conversion:**
The parse tree which is related to the source language is then converted into a Target Tree which reflects the destination language. This Target Tree or Symbol Table acts as an index to the first tree structure. The connections between the trees lie in the structures, consisting of a set of attributes that describe the links from one branch to the next. When the conversion actually occurs, it will be between two tree structures. Thus, the STENO conversion process is actually a tree-to-tree transformation.
- **Generation:**
Out of the Target Tree, the source code for the destination language is generated.

A set of language Cross-Compilers

Cross-Compilers are generated by STENO and allow the transformation of one source language into another, while maintaining the same functionality. The new source language is then compiled and executed on the new platform.

Figure 3 - ViaNova in action



Emulation

In a few cases, Cross-Compilers are not sufficient to allow a full migration of one platform to another.

For specific requirements, mostly linked with job control and Screen Management, emulators have been built for use in conjunction with the cross-compiled source code.

Wingspan

Wingspan, a window generation system for character terminals, has been designed to provide the software developer with a user interface that makes it possible for data entry forms and information windows to be painted and manipulated on a regular terminal screen.

As it is compatible with UNIX as well as PC operating system environments, the developer can offer the user a PC-like display while the investment in coding is preserved. Wingspan helps software developers achieve a consistent presentation style on multiple hardware platforms. The Wingspan family's roots lie in the Windowing Intrinsic Library with which interfaces for both PC and terminal users can be created.

Data Conversion Tools

Since all information about files and databases is automatically captured in the parse tree generated by STENO, ViaNova can also generate programs to facilitate conversion of data.

Record & Play

A “Record & Play” facility is built into the Wingspan Screen Management System. While an end user runs a program, each key stroke can be captured and recorded.

The recorded sequence can be played back subsequently as an automated test script.

Running these test scripts multiple times in parallel will allow for simple creation of stress tests. This process will also improve your understanding of your software’s performance, and help you to detect and correct performance deficiencies.

Year 2000: A Reengineering Special Case

Denkart offers a flexible and complete solution for detecting, reporting and annotating year 2000 sensitive code in your application sources. The Via2000 toolset is built using Denkart’s famous Steno Development Environment. Steno is the cornerstone of Denkart’s successful ViaNova migration solution for legacy applications. Using Steno means that the Via2000 can be adopted to any customer specific environment in the shortest time. Via2000 can be fully customized to work with many different languages and environments.

The practical problem of modifying legacy software can be enormous and insidious. This is because of the size of the code, the absence of any knowledge about the way the code works, and the pervasiveness of date-sensitive variables in the program code. Tracing down all date fields might at first appear to be a straightforward task. A simple search across the source code and data definitions should find them all relatively simple. However, *date infections* do not all show up under a superficial analysis. Hence trying to find them manually is difficult, tedious and prone to error. For this purpose, Denkart has developed a comprehensive set of tools (parser, analyzer, corrector) to address these problems specifically.

The source code analyzer, is able to identify problem date fields based on *customizable* search criteria, and can trace all of the subsequent date infections through the complete application, pointing to the exact locations where year 2000 enhancements need to be made. While not every aspect of a year 2000 conversion can be fully automated, the source code corrector is designed to provide a high level of automation, coupled with complete flexibility in implementing the required enhancements. In the first phase, the analyzer places structured comments into the source code to identify all places where a change is required, and to suggest the specific changes to be made. At this time manual intervention may take place. The developer can verify, modify and add his own structured comments when necessary. Finally the corrector engine is able to automatically execute the required changes by converting the structural comments into real source code.

Via2000 - A Technological Breakthrough

The core of their Via2000 solution is the same as for the successful ViaNova migration technology: Steno. Using Steno we can build customized language parsers and analyzers in a far shorter time frame than any of their competitors. On top of that, their Via2000 solution is complete, in-depth, and offers state-of-the-art field tracing techniques. With Via2000 you can move towards the 21st century without worrying.

Via2000 - The Unique Solution for Exotic Environments

Via2000 parses, analyses, and corrects COBOL application sources (customized extensions for SQL, CICS, and IMS are possible). The result of the analysis can be viewed in detailed reports and statistics. The analyzer also annotates the original application sources. In practice this means that comment lines are inserted where the analyzer suggests a change. These annotated sources are then fed to the corrector. The corrector implements the suggested changes. Due to the nature of the year 2000 problem, the corrector is integrated in a custom editor which allows a programmer to double check the correction before making the actual change.

Figure 4 - Defining the Year 2000 problem

What is the Year 2000 problem?

The year 2000 (Y2K) problem or Century Date Change (CDC) problem arises because of former limitations of computer technology and the historically higher cost of storing information. Most applications use only two digits to store the year. That way some storage space can be saved. Unfortunately a year consists of four digits instead of two. There is no problem as long as the year starts with "19", as in "1997". But what will happen at the end of this century when "19" changes into "20" ? How will applications which use only two digits for the year see the year "00" ? As "1900" or as "2000" ?

Often the most business critical applications will be in danger when the century changes. In the best case your program might crash,... in the worst case serious miscalculations with disastrous consequences might be the result.

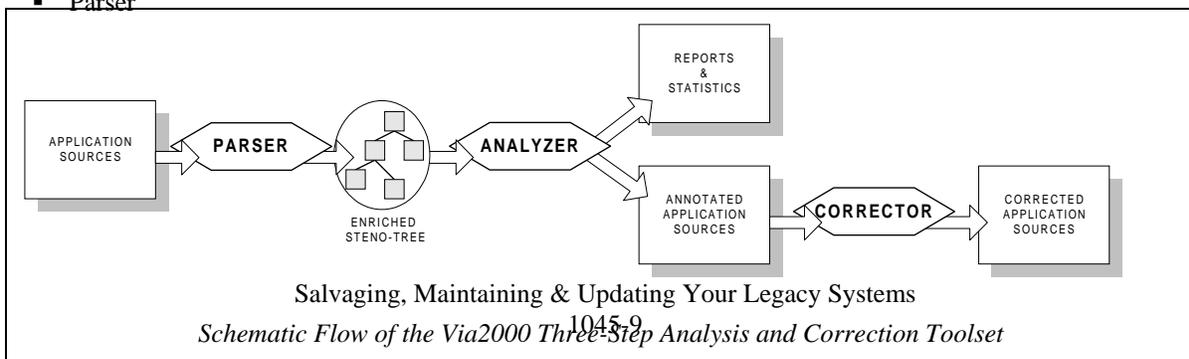
Via2000 scans your programs for troublesome fields, detects them and suggests changes. The change applied most often is to expand the year field from two to four digits. But this has to be done *before* the year 2000! For this problem there's a deadline which will not move. Companies which are not properly prepared will face law suits, loss of revenue, or even bankruptcy.

Act now, before you miss the new millennium completely!

Because of the fast development cycle using the Steno Development Environment, we can deliver Via2000 for a number of exotic and not so widely used programming languages. For example, possible target environments for Unisys are SSG, RunStreams, Mapper, and COBOL. For IBM Via2000 can be customized for COBOL, JCL, and RPG. Analyzers and correctors for other languages can be evaluated.

Technical Overview

■ Parser



The front end component is a robust syntactic and semantic parser. It processes entire applications, including sources, sub-programs, copybooks, included files, etc. The result of the parser is stored in an internal repository: an enriched Steno-tree.

- **Analyzer**
The analyzer scans the enriched Steno-tree in different passes. First it builds an inventory of possible infected objects. In a second pass it identifies any problem date fields. The identification is based on predefined and customizable selection rules. In the third phase the cross-referencing and slicing is done. Using the slicing technique the influence of one field throughout the source is traced. This way a complete list of all infected fields is built. In the end, the analyzer generates several different reports and statistics. The original sources can then be annotated; comments explaining the suggested changes are added in places where the analyzer detected a problem field.
- **Corrector**
The corrector module helps in implementing the suggested changes to the sources. The programmer is presented with a custom editor in which he/she can choose which changes to implement. After this manual supervision, the required changes can be implemented automatically.

Open-Ended Systems Corporation

OESC is a California corporation that has evolved from ConAm Corporation. ConAm is a company that has provided products and services to the Hewlett-Packard (HP) marketplace for the past 15 years. OESC has been a Hewlett-Packard DAR since December of 1995. In this role we serve as an authorized reseller for Hewlett-Packard 3000, 9000, Netserver Personal Computer, and Networking products. As of December 1, 1996, OESC has been awarded the Hewlett-Packard "Best in Class" reseller designation.

As a consulting firm, OESC has two primary areas of concentration; they are, 1) IT Infrastructure Design, Implementation and Support and 2) Application (Re)engineering including Transforming Legacies to Client/Server and Web-based Applications. Principals at OESC have many years of consulting, software and systems integration experience, which combined with our expertise, skills and resources as a computer dealer, provide us with all the requisite ingredients to significantly expand our systems integration business.

OESC is committed to providing complete information systems solutions featuring primarily Hewlett-Packard computer and networking products. In order to meet today's IT infrastructure needs, we are Novell Authorized, Microsoft Solutions Providers, Lotus Notes Authorized, a Netscape Affiliate Partner and CISCO Authorized. Hewlett-Packard certifications include sales and technical certifications for the HP 3000 and HP 9000 and HP OpenView certification.

Denkart - The Company

Denkart was founded in 1987 by four renowned computer experts. With their innovative vision and knowledge of advanced system software, Denkart built a reputation as an expert in *crunching* even the most complex problems.

The combined expertise has also lead to highly sophisticated products. *Steno*, Denkart's core technology tool, enables us to create advanced re-engineering environments. By using *Steno* we can produce customized migrations to open systems for your legacy applications.

Another innovative product of Denkart is *OpenMessage*, an integrated message handling system for fax and telex traffic.

The offices in the USA and Belgium continue to expand while Denkart builds on *The Denkart Way*.