Legacy Value Restoration

500 Glenpointe Center West
Teaneck, NJ 07666
www.cognizant.com
Introduction

Background
The emergence of eBusiness systems, CRM applications, and Data Warehousing solutions has forced IS organizations to develop coherent plans to evolve their legacy technology bases to support the IT demands of new business initiatives.

Most IS organizations face the following problems with their existing systems:

• Should they be minimally maintained while the organization develops new systems?
• Should they be replaced by packaged software?
• Should they return to their previous level of enhancement/maintenance?

Legacy systems contain useful business knowledge, but extracting that value is becoming increasingly difficult. The evolution of today’s systems must begin with their replacement within the context of business decisions. Reuse of legacy rules allows businesses to gain significant time-to-market advantage. Inappropriate reuse results in project failures.

Reuse of legacy systems, data, and business rules are important in the current economic conditions.

Objectives of the Document
This document explains the importance of legacy systems in the current day context and how one can reposition and leverage their existing legacy technology foundation to meet the current and future business use.

In addition this white paper describes the Cognizant approaches, methodologies, and its proven processes in restoring the legacy applications that are developed using various technologies that require specific answers.

- Web Portal Development
- Development of Knowledge Management Systems

Organization of this Document
The next section provides an overview of the legacy restoration importance, and phases involved in the process.
• Legacy understanding, its scope, various layers and relationships with other restoration phases.
• Data profiling.
• Various techniques in legacy application extensions.
• Legacy Transformations.
• Client Server systems.
• Cognizant Legacy Restoration process.
• Cognizant approach to legacy systems.
• Appendix A, B, C, D, and E lists the various third party tools available to help Legacy understanding, Data profiling, Legacy Extension, Legacy transformation, and C/S environment respectively.
• Appendix F describes the capabilities of Cognizant’s Legacy transformation and understanding tool- COBSAT.
• Appendix G describes the technology used in building the COBSAT.
Restoration of Value of Legacy Systems

The forces driving the interest in restoring legacy systems come from outside the usual Application development organizations. Organizations must prepare themselves for this so-called continued business pressure and learn to assemble applications while satisfying time-to-market demands.

The primary drivers of restoring legacy systems include:

- Extending the reach of traditional, internally-directed business processes to external constituents.
- Using external sources of application development such as packages or out-sourced development.

E-business and CRM initiatives open up traditional "closed" business processes to an organization's external buying and selling relationships. Increasing the speed of delivery of inventory is vital for suppliers, but when it comes to managing the flexibility of goods in an increasingly customized world it is equally important to sellers. Offering personal customization to clients and business partners increases the service component of any company's product. However, the traditional processes of running a business still exist and their technological implementations are still active. Leveraging these technology investments in the more demanding world of e-business is a challenge faced by many organizations.

The technological difficulties of integrating legacy systems together with new development or application packages are many.

Different application paradigms, unstructured and unorganized systems with few design characteristics for integration, the constant allure of the new over the old, declining skill sets, and the technological incompatibilities make integration of legacy systems more complex. But still organizations continue to use expeditious, yet shortsighted, approaches when it comes to integration that provide short-term gain and potential long-term problems.

Restoring the legacy systems should be treated as an evolutionary process. So, why leverage legacy systems? Because they signify the existing legacy business processes. To accept that packaged solutions or new development can re-implement these processes with the same level of accuracy and specificity, reducing either short-term or long-term costs is presumptuous. Cognizant does not propose that mainframes are the correct platforms for all new development. Some applications need the quality of service characteristics of this platform and others do not. Restoring legacy systems is about connecting the ones that do with the ones that do not.
Market Statistics

According to analysts, more than 180 billion lines of code in hundreds of languages populate the operational inventory of commercial IT systems. Much of the inventory is in one of more than 80 variations in COBOL with PL/1 and RPG being the next most common area. The balance is filled by hundreds of other languages, proprietary and standards driven, procedural, event driven or object oriented. According to Gartner, 40 percent of all mainframe applications will be modernized to make their functioning available for the construction of composite applications by 2003.

Phases of Legacy Restoration

The evolution of today’s legacy systems ranges from relatively simple to extremely complicated. Legacy system goes through the following phases, before fully integrated with the new systems:

*Legacy Understanding*: The ability to isolate business functions to technological implementations. Finer granularity requires greater levels of understanding.

*Legacy Extension*: Extending access to legacy transactions or data with minimal modification to existing systems.

*Legacy Transition*: Wrapping or transition of Legacy systems to new languages or technology environments.

Legacy Restoration Areas

According to Cognizant the following phases or technology solutions will play a major role in restoring the importance or usage of the existing Legacy systems:
• Legacy Understanding
• Data Profiling
• Legacy Extensions
• Legacy Transformations
Legacy Understanding

The need to understand existing systems has traditionally been more reactive than proactive. When an organization opts for a maintenance change, it develops the level of understanding required to introduce the change and sorts out the rest in testing. Academic research has shown that as much as 25 percent of maintenance costs results from high levels of program and system complexity. Developing more strong levels of understanding is the first step toward reducing both complexity and maintenance costs. Similarly, organizations should follow Cognizant process in understanding their legacy systems, to make their systems up to date.

Peeling the Layers

Understanding existing systems occur at different levels of granularity and specificity. Like peeling an onion, each layer brings you closer to the core. From bill-of-material-like relationships, to detailed program understanding, procedure flow or data flow, organizations should peel each layer separately and use the correct level of detail necessary for the job.

Application Artifacts

Cognizant use the term "artifact" to represent the various pieces needed to develop applications, which it has taken from SEI CMM process. Cognizant use this term, rather than "objects" or "components," to eliminate confusion with the other meanings of those terms. Application artifacts in an IBM mainframe environment would include Database Descriptors, Program Specification Blocks, Message Formatting Services, Screens, copybooks, and file definitions. These represent the artifacts associated with system-level understanding.

Application understanding layers can be categorized as:

- Applications and their relationships
- Application Control and Data flow and Business modules
- Process Flows and Business rules

Applications and their Relationship Layer

In the process of understanding legacy systems, organizations first can discover

- Artifacts they did not know they had
- Relationships they did not realize existed or Artifacts no longer in use.
The level of understanding establishes high-level relationships and application interfaces that are useful while replacing an existing system with a software package. It identifies the transition points between legacy applications and the new system. This level of understanding requires recognizing the relationship between application building blocks. Documenting and maintaining relationship information is easier within a homogeneous (e.g., mainframe) environment. Maintaining relationships between components across heterogeneous application environments is difficult.

**Application Control and Data flow and Business modules Layer**
The next layer of understanding involves studying the program code. By analyzing program code in combination with artifact relationships, an understanding of control flow and data flow is developed. This level of understanding is necessary to create large-grain components that may be used in conjunction with latest software implementations, particularly where business models require a blend of the new with the old.

**Process Flows and Business rules Layer**
Finally, understanding inter- and intra-process flows is important to extract business rules and develop independent single-function components. If the programs are highly complex, it could be difficult or even impossible to interpret.

**Program Complexity**
Application complexity plays a major role in understanding a given legacy system. An objective measure of program complexity is represented by the software measure, Cyclomatic complexity. Cyclomatic complexity represents the number of independent paths through a program. The Software Engineering Institute (SEI) provides the following continuum for identifying complex programs:

<table>
<thead>
<tr>
<th>Cyclomatic Complexity</th>
<th>Risk Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 10</td>
<td>A Simple Program without much Risk</td>
</tr>
<tr>
<td>11 – 20</td>
<td>More Complex; Moderate Risk Program</td>
</tr>
<tr>
<td>21 – 50</td>
<td>Complex; High Risk Program</td>
</tr>
<tr>
<td>&gt; 50</td>
<td>Un-testable Program (i.e. Very high Risk)</td>
</tr>
</tbody>
</table>

Source: SEI CMU
This level of understanding becomes even more critical when extraction of distinct business rules and transforming to new languages and platforms are required.

Cyclomatic complexity plays an important role in the legacy evolution process.

**Scope of Legacy Understanding**

**Major Components of Legacy Systems**
At a high-level point of view, Cognizant divides application architectures into the following three major components:

- Application presentation
- Application logic
- Application data

All these components play a major role in identifying the business rules in a given application. Application logic components play an important in process flow, data flow, and application control.

Many GUI extension products enable a more palatable and accessible presentation via the Web for many enterprises. Isolating the presentation logic enables the various enterprises to increase the sophistication of their integration capabilities.

Further isolation of business logic offers the potential for components that can be reused during application integration initiatives.

Basically understanding the data is not considered as a part of legacy system study. Due to various historical reasons legacy data requires profiling, which includes refining.

Legacy restoration phases require various levels of understanding of the legacy systems. The following sub sections describe the level of understanding required for each phase:

**Legacy Extensions**
GUI Extension approach will help enterprises to have less need for increased levels of program understanding. Various host-to-web products improve the front-end of the existing systems.

**Legacy Integration**
Current system and program understanding knowledge at most enterprises are not sufficient to implement these increased levels of evolution. Whether driven by application integration needs, map/gap analysis for packaged software replacement or identifying interface points to new software packages, a robust understanding of application data and control flow is mandatory.
Legacy e-Business Culture Clash
Most enterprise application integration initiatives involve some level of Java implementation. Understanding system flow, interfaces and data relationships in an environment such as MVS COBOL/CICS/DB2 increases the need for automated help.

Leading vendors have a propensity to support multidiscipline technologies in an integrated fashion. Some vendors did make an attempt to meet the demands of the new e-business drivers and application integration initiatives.

Application Restructuring
Simply reducing the cost of maintenance by sinking the system complexity has never been a compelling drive for most companies. A significant piece of the future demands on AD will be integration.

Cluttering the presentation layer with business logic and data navigation logic causes most developers to shy away from restructuring the ravages of time.

How to restructure Legacy Application
Extracting components from legacy systems requires reasonably structured code. Without objective software metrics, rationality is difficult to determine. Use these metrics to determine the extremes and exercise care with complex legacy systems.

For programs with high-complexity metrics, Cognizant suggests enterprises wrap them "as is," rather than attempt to extract reusable components.

Greater levels and finer granularity of understanding will reduce project duration as well as the need to create larger-grain and less-reusable components because of misunderstanding.

Much academic effort has been put into understanding as the basis for extracting components or creating wrappers. The literature reflects results with simple programs or well-structured code.
Legacy Data Profiling

Global 2000 companies have deferred strategic business system initiatives for years while they grappled with massive Y2K upgrades. Now, these companies are now turning their attention to high-priority projects such as Customer Relationship Management (CRM), e-Commerce, and Data warehousing.

Data profiling is used with various names such as online profiling, business intelligence, Data quality Management and analysis of data for application stability. This note restricts the scope of data profiling to Legacy data profiling. Traditional data profiling, which is a part of Data Quality and Management provides consistent and clean data for various application systems.

Data Profiling: What is it?

Data profiling is a metrics-based, systematic assessment of data. The specifics vary from vendor to vendor, but in general, measurement algorithms quantify content, structure, sizing, and usability. The purpose of these measurements is to audit non-conforming data. Dirty data (known as non confirming data) is what cleansing and transformation tools or custom scrubs can fix. The trick is, you have to find the dirty data before you can fix it and data profiling finds flawed data.

Many IT shops do a few pieces of data profiling with manual analysis or custom queries. The problem is that a manual process requires time and patience, as well as tolerance for gaps and errors. Because of the time and costs involved, data analysis usually is done on a spot check or sampling basis. And what usually happens is an application of the 80-20 rule: 80% of the effort gets devoted to fixing the 20% that slipped through the samples and checks.

Automated data analysis: Data Profiling can be done on every attribute in every record, without human error. This means that you are able to find all dirty data at once and develop a proactive cleansing strategy.

Data Profiling Technologies and their Status

Data Profiling is a mature technology and is a part of Data Management, Data Warehousing, and ERP system implementations. ETL tools help to profile the data to some extent, which is required to build the data base systems for analysis.

Legacy data profiling technologies helps to analyze the data in terms of

- Inferred data type for each column or field
• Analysis of data in all fields or columns
• Inconsistencies with respects to data characteristics and relationships
• Data relationship building
• Data normalization

As a part of data profiling, one has to build the data repository and data transformation or mapping to either legacy data or new data model from repository.

The technology is matured and is very relevant due to the explosion of e-Business and CRM applications.
Legacy Extensions

Legacy extension solutions originated in 1997 because enterprises wished to provide access to traditional legacy systems to untraditional users — i.e., to users external to the enterprise. Converting 3270 or 5250 character-based data to a form more meaningful and accessible to constituents unfamiliar with the trappings and peculiarities of this presentation or business paradigm has become commonplace. What has changed is the variety of sources of business information now being made available outside traditional channels, as well as the variety of interface alternatives.

As these new possibilities drive changes in workflow, the need to modify existing business processes as implemented in current systems can exceed the ability of most host-to-Web products. Using 3270 data streams as an API has its own limits than HTML, which is designed for the presentation of documents. Attempting to leverage existing business processes in new ways require modification of existing systems. Even the use of software wrappers, which minimizes the change to existing systems, does not eliminate completely.

Software Wrappers

Software wrappers represent a bridge between traditional procedural programs and Object Oriented (OO) methods. Procedural programs were written to implement a particular application function, such as processing orders. In order to fulfill this function, they read and update a number of different data sets. The actions that take place on the data are carried out in a procedural order, which code occurs in the program. The program is a representation of the process and not the data. OO describes the processes or methods, in terms of the object or data. While there are several types of software wrappers, object wrappers are designed to use the existing procedural code as an extension of an object oriented method. The object wrapper operates as a called method of an object, and then uses a traditional procedure call to execute an existing application function.

Black-Box Approach

Extending the business logic requires the use of wrapping technology or code transition tools. The granularity of a wrapped capability depends on the ease with which the capability can be isolated. Batch processes, transactions, programs or even subroutines can be wrapped for use by new applications. These approaches minimize the granularity of understanding needed. This "black-box approach" isolates internal complexity that has suffered the ravages of time.
**White Box Approach**

Organizations can leverage the business rules of existing systems, but remove themselves from the limitations of the application environment. Transitioning existing business function to new environments requires the understanding with wrapping and the ability to translate languages and calling parameters. Understanding business functions and identifying the data items that represent these concepts, as well as the processes performed on them, are critical. This "white-box approach" requires a more detailed level of understanding, but will generate finer-grain components potentially of greater reuse value. This is a longer-term strategy, rather than the short-term tactical nature of the various extension approaches.

**Transaction Wrappers**

Extracting information from legacy systems for presentation through the Web or composite applications is common today. Moving from straightforward interactions with existing systems by new constituents, to the more traditional transactions of legacy TP monitors, requires more programmatic connections to ensure integrity. Software wrappers do enable existing business logic to be reused unchanged, but does not preclude software changes to existing programs. The interface logic usually requires modification to accept input parameters. When using Java on OS/390 to develop wrappers, the data type and parameters passing between these wrappers and legacy require the use of the Java Native Interface and a C/C++ intermediary program.

Cognizant have identified three classes of legacy extension functionality:

- GUI extension
- Transaction extension
- Data extension

**GUI Extensions**

Because business drivers are pushing to move applications to the Web, many enterprises are considering legacy extension as a cost-effective way to take existing applications to the Web, or to create new hybrid applications that tactically use legacy systems' functionality.

Many organizations which are looking to get to the Web as quickly as possible find that extending the GUI is a "quick and dirty" way to put more life into an existing application. There are several approaches in achieving this goal. The major ones are:

- One includes middleware-only solutions that interpret screen definition languages for the 3270 mainframe, 5250 AS/400, VT100 DEC and HTTP formats. Middleware-only solutions may also pass an interpreted screen or a standard data stream.

- Another approach includes extended functionality added by a number of leading vendors to increase value by suggesting a windowed approach to the resulting screens (sub-screens).
Yet another approach features value-added languages that will allow easy creation of new workflows by combining and dissecting screens that can be customized around specific events and clients.

**GUI Extension Requirements**

- AD development language
- Semi-automated screen capture
- Screen rule/template capture
- Open interfaces
- Shared development
- Many-to-many screen mapping
- Dynamic value-added graphics
- Key and cursor control
- Deployment requirements
- Logging and monitoring capabilities
- Multiple high-volume server options
- Multiple input stream supports

**Transaction Extensions**

Business data from legacy systems that is used on the Web sites is often obtained through direct data access (ODBC/JDBC) or 3270 access. By emulating a 3270 terminal in a middle tier, IS organizations can access legacy data through traditional transactions, then include this information in a browser-based metaphor that is more acceptable to external constituents.

As IS organizations become experienced at enabling increased access of traditional systems to new and varied constituents, the use of the presentation layer as an integration point becomes fraught with danger. Increased complexity, duplicate business logic, maintenance headaches or data inconsistencies become difficult to manage.

As traffic increases, scalability and manageability become larger and larger hills to climb. For many enterprises, a return to two-tier implementations is plausible.

In these instances, CICS can be seen as the point of integration rather than simply as a means of accessing data using traditional transactional programs.

IBM continues to evolve CICS Transaction Server to enable connectivity of CICS transactions to many non-CICS processes. In order for the enterprise to take full advantage of these capabilities while also leveraging existing programs, it must separate communication logic from the remaining business logic to ease implementation. Most enterprises have chosen to postpone this effort and use the existing 3270 logic as an API, while doing so may satisfy demands for immediate Web access and are desirable only for the most-direct interactions. The continuing evolution of organization structures as a result of the possibilities offered
by technology should change the way companies think of the traditional presentation layer. The loss of control of the recipients changes the demands on this programming effort. While many products offer more sophisticated integration means using 3270 as the API, but this is not the long-term solution.

**Transaction Wrappers – Issues**

Some of the major issues in building the transaction wrappers are:

- Session Management
- Performance and Scalability
- Wrapper building

**Sessions Management**

One of the technical difficulties associated with legacy extension solutions is rooted in the fundamental differences between the session-based paradigm of legacy applications and the session-less paradigm of the Internet. The transaction wrapper solution must commit a session to client interaction, to maintain the integrity of the application, session, and the data.

**Performance and Scalability**

The transaction wrapper solution must commit a session to client interaction, thereby severely limiting scalability, or manage a dynamic pool of sessions against an ever-changing source of Internet-based interactions. In addition, the solution should provide clustering and load-balancing capabilities to give the legacy applications the required scalability as their needs grow.

**Wrapper Building**

Software wrappers implement methods through a more traditional "call" to an existing procedural program. Depending on the granularity desired and the current implementation of the legacy program, re-engineering the interface may be called for. Wrappers can exist, in increasingly finer granularity, for entire batch jobs, existing online transactions, programs, program modules, and procedures.

Program modules are probably the easiest to wrap, or encapsulate, because they are already designed as independent callable modules. Encapsulating program procedures is the most complicated one, since it generally requires restructuring an existing program to enable internal procedures to be externally called with the appropriate parameters.

The following guidelines should be followed in building the wrappers:

- Re-engineering entire programs around the Object Oriented paradigm is not a recommended solution.
- Unstructured code must be corrected, to (more clearly) isolate, code segments with procedures.
- Code-slicing tools are needed to isolate business logic. Object definition is not obvious from historical procedural implementation, although persistent data stores offer a good starting point.

- Further, redundant and dead code should be eliminated to form more crisp program code.

- Finally, name rationalization is necessary, in order to more cleanly associate attributes with the resulting methods.

- One of the keys of Object Oriented implementation is the isolation of data fields only to the methods that use them. Traditional procedural programs lump all fields with either file definitions or working storage. In many cases, the same field is used by different code segments, which result in different methods. These field usage problems need to be understood to enable proper isolation.

Legacy Data Extension

Legacy integration is fraught with problems of diversity such as diversity of data sources and client types. Legacy Data extension tools can be used to separate the data source interface layer and the client interface layer from the integration transaction server capability. This enables continued growth of data sources and clients. The major Legacy data sources include:

- 3270
- 5250
- Virtual Terminal
- SQL
- MQ Series
- COM
- External applications

Data Understanding and Recovery

Various Legacy extension technologies allow the understanding of the implied data models in the existing physical files. These physical models can be turned into logical models for further enhancements/extension at one end of the spectrum and into different file definition languages. Many of these vendors use repositories for downstream uses. Another set of vendors in the data area allows the extended use of the actual data for either downstream reporting or for transporting of data to new environments and new integration opportunities.
Legacy Transformations

Some enterprise software projects reflect the demands of longer-term business strategy; others are aimed at meeting an often-urgent need of a particular business demand. To satisfy both needs, IS organizations must develop coherent plans to evolve their technology base to support the IT demands of immediate and long-term business initiatives.

Although legacy systems contain useful business knowledge, extracting that value remains a problem. Many organizations have little desire to open up a "Pandora's box" of legacy systems.

Current trends indicate noninvasive extension approaches are preferred. Extending existing applications to new environments is the least risky and often the least-invasive approach. It provides immediate short-term, and low-risk resolutions to immediately address the demands of e-Business for open access from traditionally internal applications to new external constituents.

Scope of Legacy Transformations

The legacy transformation market consists of those enterprises looking for tools that provide business rule identification, code slicing, and code modification or transformation from one language to another. These products are generally presented as add-ons to legacy understanding tools, which operate on a mainframe or a workstation. Other sophisticated tools also support language wrapping for creating components out of legacy systems and provide support for porting legacy business logic to new architectures and languages.

The risk of completely transforming an existing application to a new platform, limits this choice to those with the oldest platforms. These "burning platforms" are beyond hope of continued growth and evolution.

Many enterprises are looking to transform an existing system for reasons other than concern for the current platform's longevity. In some cases, these enterprises may be driven by concern about availability of dwindling resources, such as COBOL or Natural programmers, and IDMS or assembler language skills. This transformation discussion is about redeveloping a mainframe-based application that was implemented in a number of traditional legacy languages to a Java object-oriented platform. In between these two approaches are more-invasive solutions that provide programmatic interaction in a multi-tiered environment.
Client Server Systems

Only a decade ago, the term "client/server" identified a whirlwind of new products and architectures for IT. Just as with mainframe legacy applications, these client/server applications now need to be included in a continuing assessment of your IT portfolio. High support costs, inflexible design and waning vendor support will require you to rebuild many of these applications, but the business process design and user knowledge can be useful starting design points for the replacement systems.

Scope

Legacy applications are those that you have in use. Regardless of their age, applications in production by definition have business value. The task is to identify and balance cost and life span for the organizational good. Client/server applications may be centrally managed or distributed in work groups. The common thread is that they use PCs for client code and some sort of persistent communication to a server that runs either a database or a database and an application.

Client/Server Application Artifacts

Cognizant uses the term "artifact" (borrowed from SEI CMM) to represent the various pieces needed to develop applications. Cognizant uses this term (instead of objects or components), to eliminate confusion with the other meanings of those terms. Application artifacts in a client/server or distributed environment would include, for example, programs, methods, classes, databases, stored procedures, triggers, and packages. These represent the artifacts associated with system-level understanding.

- SQL objects
- Objects inside and outside a package
- Objects belonging to different databases
- VB/COM objects (forms, controls, VB classes, COM classes)
- Web objects (JSP/ASP, HTML, scripts)
- Oracle forms, PowerBuilder and Java

Targeted impact of change searches are provided with the product. For example, determining the impact of changing a column size in an SQL database can be accomplished through drop-down selections. Results indicated impact on database and related files, as well as programs, stored procedures, etc. A more generic search capability is also provided, enabling "what if" searches across the entire knowledge base.

Cast provides the market-leading tool solution for understanding in the client/server and distributed environments.
Legacy Restoration Process

Cognizant’s legacy restoration process studies the existing systems in a systematic way and provides comprehensive solutions to customers.

Phased Methodology

Cognizant recommends the following process, with well-defined phases. This phased process describes, how Cognizant approaches handling legacy systems:

System Inventory
As a part of this process Cognizant first identifies and lists all the relevant customer legacy applications and business processes. As a part of this legacy inventory collection, if possible relationships will be developed between the processes and applications.

Value Measurement
In the next phase Cognizant attempts to find answers to the following questions, to evaluate the importance of the given legacy application for the customer future IT strategy:

- Does the product/application still do the desired job and also meet the new business requirements?
- What are the anticipated changes in the business that will change the way the product/application is used or the volume of business it handles?
- Does the product/application provide the required information for integration of new applications?
- Will the product/application scale up to meet the new business requirements?
- How would the work be done if the legacy solution should disappear?

The above questions will help to identify the portion of the legacy portfolio, which continues to be of business use. The next step in this phase is to identify the high-risk applications. Three issues dominate evaluation of the legacy application portfolio on any platform:

- The remaining life of the platform
- The availability of appropriate skills
- Technology

Platform Assessment
Is any part of the application a "burning platform"? In other words, are there hardware or software components that are or soon will no longer be formally supported? In the client/server world, the client development or runtime, the middleware and the database must all be included in the assessment. Business-critical applications should not run on unsupported platforms. If the platform is unsuitable, maintaining it "as is" ceases to be an option and the decision should be made to move or replace.
Enterprise Technical Skills
Does the organization/enterprise have the skills to maintain or enhance the given application? As tools age, training courses become less common, and staff is reluctant to be locked into aging and unpopular skill sets. An example is provided by client technologies: Visual Basic remains popular and well supported, but tools such as Gupta and Progress 4GL are now much less popular. Even within Visual Basic, the version-to-version changes may make the code obsolete.

Evaluate the Technology’s Health
Even if the platforms are sound and skills are still at hand, the architecture or structure of the application may condemn it to a short life. If the code can be separated into logical tiers user interface, business rules, and data access then each tier could be modernized or altered independently. Consequently, poorly documented code or complex implementations are more easily replaced than preserved or extended. The health assessment of the technology asset needs to answer the following questions of application portfolio:

- Who are the developers?
- Who maintains the code?
- Is there documentation, and does that documentation describe anything that resembles the product as it is today?
- What products were used in development and are necessary to operate the application?
- Are they still available?
- Are they back-level?
- Are they under license?

Others
Other technology issues are scalability and security. Application system design presumed persistent connections and usually internal users. If the product is extended to the Web, then both these assumptions can fail and would require substantial redesign.

Case Studies
The following Case studies are the examples of extreme cases that illustrate the use of this process.

Case Study 1
A multtier client/server application. Thin client or clearly structured client built in Visual Basic. Relational database management system in the server (from Oracle, Microsoft or IBM), server code in C, little use of stored procedures. Data model organized to isolate critical or proprietary data into tables separate from transaction data. Decision: Maintain, extend and enhance.

Case Study 2
Fat-client, built-in proprietary tool by third party. Its back end uses flat files or proprietary indexed access. No clean delineation between client user
interface and logic, or server logic and data access and there is no data model. Proprietary information stored in the same tables as transaction data. No licenses to development environment available. Documentation in the form of the original request for proposal only.

Decision: Analyze business processes and user interfaces as design guide for a replacement.
Cognizant Approach

In addition to identifying the value of a given legacy system(s), in a systematic process, Cognizant developed processes to understand, extend, data profile, and transform legacy systems.

Legacy Understanding Approach

Data Profiling Approach

According to data quality experts, when significant quality problems exist in source data, data profiling should first be used to analyze and identify problem areas.

Conventional Techniques in Data Migration

The conventional approach to data profiling and mapping starts with a team of professionals (data and business analysts, data administrators, database administrators, system designers, subject matter experts, etc.) who meet in a series of joint application development (JAD) sessions. They attempt to extract useful information about the content and structure of the legacy data sources by examining outdated documentation, COBOL copybooks, inaccurate Meta data, and in some cases, the physical data itself.

Problems and Pitfalls: Profiling legacy data in this way is extremely complex, time consuming and error-prone. After the process is complete, only a limited understanding of the source data is achieved. At that point, according to the project flow chart, the data analyst moves on to the mapping phase. However, since the source data is so poorly understood and inferences about it are largely based on assumptions rather than facts, it results in an inaccurate data model. Based on this information, the data is extracted, scrubbed, transformed, and loaded into the new database.

Problems and Pitfalls: Development of an inaccurate data model and set of mapping specifications. Not surprisingly, in almost all cases, the new system doesn't work correctly the first time. Then the rework process begins: redesigning, recoding, reloading, and retesting. At best, the project incurs significant time and cost overruns. At worst, faced with runaway costs and no clear end in sight, senior management cancels the project, preferring to live with an inefficient but partially functional information system rather than incur the ongoing costs of an "endless" data migration project.
**Problems and Pitfalls:** Senior management cancels the project, preferring to live with an inefficient data model.

**Solution:** The answer is a new category of software called data profiling and mapping which offers a fast, accurate and automated way of understanding data. It enables a small, focused team of technical and business users to quickly perform the highly complex tasks necessary to achieve a thorough understanding of source data. This level of understanding cannot be achieved through conventional approaches.

Data profiling and mapping software enables data migration and integration projects to be completed successfully the first time, eliminating extensive design rework and late-stage project cancellations. It can even warn IT management when the business objectives of the project are not supported by the data, thereby dramatically lowering project risk and enabling valuable resources to be re-directed to other, more fruitful projects.

**Strategies for Data Migration with Data Proofing**

Data profiling and mapping consist of six sequential steps, three for data profiling and three for data mapping, with each step building on the information produced in the previous steps. The resulting transformation maps, in turn, can be used in conjunction with third-party data migration tools to extract, scrub, transform and load the data from the old system to the new system.

Data sources are profiled in three dimensions: down columns (*column profiling*); across rows (*dependency profiling*), and across tables (*redundancy profiling*).

*Column Profiling:* Column profiling analyzes the values in each column or field of source data such as inferring detailed characteristics for each column, including data type and size, range of values, frequency and distribution of values, cardinality and null and uniqueness characteristics. This step allows analysts to detect and analyze data content quality problems and evaluates discrepancies between the inferred, true Meta data and the documented Meta data.

*Dependency Profiling:* Dependency profiling analyzes data across rows by comparing values in every column with values in every other column. It infers all dependency relationships that exist between attributes within each table. This process cannot be accomplished manually. Dependency profiling identifies primary keys and whether or not expected dependencies (e.g., those imposed by a new application) are supported by the data. It also identifies "gray-area dependencies" those that are true most of the time, but not all of the time, and are usually an indication of a data quality problem.
**Redundancy Profiling.** Redundancy profiling compares data between tables of the same or different data sources, determining which columns contain overlapping or identical sets of values. It looks for repeating patterns among an organization's "islands of information" such as billing systems, sales force automation systems, and post-sales support systems. Redundancy profiling identifies attributes containing the same information but with different names (synonyms) and attributes that have the same name but different business meaning (homonyms). It also helps determine which columns are redundant that can be eliminated and which are necessary to connect information between tables. Redundancy profiling eliminates processing overhead and reduces the probability of error in the target database. As with dependency profiling, this process cannot be accomplished manually.

Figure 1: Key steps in data profiling and mapping

After the data profiling process is complete, the profile results can be used to complete the remaining three data mapping steps of a migration project, they are normalization, model enhancement and transformation mapping.

*Normalization:* By building a fully normalized relational model based on and fully supported by the consolidation of all the data, the data model will not fail.

*Model Enhancement:* This process involves modifying the normalized model by adding structures to support new requirements or by adding indexes and denormalizing the structures to enhance performance.

*Transformation Mapping:* After the data model modifications are complete, a set of transformation maps can be created to show the relationships between columns in the source files and tables in the enhanced model, including attribute-to-attribute flows. Ideally, these transformation maps facilitate the capture of scrubbing and transformation
requirements and provide essential information to the programmers creating conversion routines to move data from the source to the target database.

Developing an accurate profile of existing data sources is the essential first step in any successful data migration project. By executing a sound data profiling and mapping strategy focused technical teams and business users can quickly perform the highly complex tasks necessary to achieve a thorough understanding the source data.

**Better Techniques**

By following these six steps in data profiling and mapping, companies can take complete their data migration projects successfully. Data profiling and mapping, if done correctly, can dramatically lower project risk, and enable valuable resources to be redirected to other. Finally, it will deliver higher data and application quality, resulting in more informed business decisions.

As Global 2000 organizations turn their attention to CRM, e-Commerce, data warehouses and other major new initiatives, Data profiling and mapping is becoming an increasingly important solution for integration and data quality problems. Understanding the source data means that these strategic projects can be completed on time, on budget, with less risk of failure.

**Legacy Extension Approach**

According to estimates by the Meta Group, 60-80% of corporate data currently resides on mainframes. Other studies indicate that the cost of managing/administrating users and data on a mainframe is much lower (2 to 3 times) when compared to UNIX servers. The other major advantages of keeping applications and data on a mainframe are:

- Security -- access control, auditable
- Proven performance
- Reliability, serviceability, and availability
- Known environment for years: Trained users and support personnel.

Using web technology, the existing mainframe applications and data can be made available to users without sacrificing any mainframe feature. The following subsections discuss the methods to make the mainframe information available on the web. Depending on the requirements, the web can be categorized as Internet, Intranet, or Extranet.

**Web Enabling Techniques**

There are four techniques for web enabling mainframe applications:

- Native Mainframe Web Server
- Screen Scraping
- Distributed Application Systems
- Data Replication
Native Mainframe Web Server: In this web enabling approach, the native web server and the native or third-party products are used to deliver the existing application 3270 screens into http dynamic web pages. Application input and output (3270 data stream) is routed through a Web server instead of a terminal session. TCP/IP on the mainframe is used as the transport mechanism. This approach is 2-tier and does not need a middle tier of hardware, software, and administration. The existing mainframe systems can support thousands of simultaneous users.

The following products can be used for this native mode approach:

- CICS Web server
- CICS Internet gateway

Advantages
- Mainframe security is leveraged including the existing access control system.
- Excellent performance on database queries and transactions can far outstrip a 3-tier solution.
- Provides unmatched bandwidth and I/O capabilities.
- Logon session overhead can be eliminated.
- Reliability, scalability and availability are guaranteed.
- Gradual client/server conversion of applications (if applicable) is possible.
- Uses native data store interfaces (DM2, IMS, VSAM, etc.)

Disadvantages
- Mainframe coding is required to re-route input and output from the applications.
- Mainframe programmers need to be trained on web principles.

Screen Scraping: In the screen scraping approach, a gateway server consists of a CGI running on UNIX or NT based Web server. CGI maintains multiple terminal sessions to the mainframe and interacts with those sessions, translating 3270 screens to HTML. The gateway server then sends the HTML screens to the Web users. This three-tier architecture based solution consists of clients (web browser), middle layer (web server + 3270 to html converter), and application server (IBM Mainframe).

The companies that provided these solutions were earlier selling proprietary desktop software that scraped and beautified a terminal session. But because Netscape and Microsoft gave away their Web clients and these companies did not, they are now in the business of beautifying terminal sessions on an intermediate Web server.

Products such as ResQ!Net, Jacada can be used instead of CGI scripts. ResQ!Net and Jacada will convert 3270 applications on the fly to HTML pages. These products also provide features to beautify the 3270 screens.
Advantages
- Application code on the mainframe remains untouched.
- Added capabilities of UNIX or NT based Web server (such as server toolkits) can be leveraged.
- Has a low startup cost.

Disadvantages
- Mainframe capabilities remain unleveraged.
- Terminal sessions require additional support.
- Requires the system overhead of a logon.
- Applications cannot be extended.
- Because of logons to terminal sessions, the Web browser cannot use the browser navigation buttons. Users must be instructed to disable their Web browser's cache, not use forward and back buttons, and not reload the screen.
- Performance is slower than an equivalent terminal session
- Has a high rollout cost.
- Requires a user ID and associated administrative overhead for each user.

A screen-scraping project can be started quickly. With just a single server, some software, and a little bit of time coding workstation scripts, mainframe applications such as time sheets and purchase requisitions can be made web accessible.

Rolling this pilot out for thousands of users requires more time, money, and risk. Another drawback is the creation of a dispatching infrastructure on top of the dispatching capabilities of the mainframe. The architecture of the middle layer will be based on the load. Very few corporations can predict capacity requirements while dealing with a variety of internal and external users. But the middle level UNIX or NT servers can be added dynamically depending upon the load.

In conclusion, while there are some benefits to screen scraping in some applications, it is not a good long-term solution for majority of the organizations.

Distributed Applications
In this approach, the application is re-developed for web to access the mainframe applications and data. The approach follows object oriented methodology. In this approach, the distributed objects are developed around mainframe applications and data by putting wrappers around them. These distributed objects can be accessed either directly from the client (browser) or from the middle layer server. Based on the requirements and readily available of a given technology, this architecture could be a 2-tier or 3-tier.

In this approach, there is no need to touch the applications or the data on the mainframe. This technique relies on the tools such as AS/400 toolbox
or S/390 Java API libraries. In addition, this model can also make use of the existing C/370 API libraries for CICS, VSAM, and DB/2.

A number of third-party libraries, such as CA ODBC interfaces to various IBM file systems and databases, are available to help implement this approach. The architectural schema for this solution is identical to screen scraping, but the software components are different:

**Advantages**
- Mainframe code is left untouched.
- The added capabilities of workstation web server (server toolkits) can be brought to bear.
- There is no (continuous) session overhead on the mainframe.

**Disadvantages**
- The periodic updates of data must be made across the LAN, and are limited to LAN (rather than mainframe bus) speeds.
- The addition of web servers increases the maintenance burden.
- Workstation reliability becomes an issue. Servers will have to be periodically re-booted to prevent memory leakage.
- System does not support Online Transaction Processing (OLTP). Coding is required on the UNIX or NT server to duplicate the business rules from the mainframe.

**Data Replication**
In this approach, the application is developed for the web to use the replicated mainframe system data. The new application system and the replicated data are in the middle layer. Data across environments (i.e. middle layer database and the mainframe database) can be coordinated with periodic database queries and batch updates.

This technique relies on data and application replication. It requires setting a gateway server using remote data access protocols such as ODBC, SQL*Net, and DRDA, which provides the link between the mainframe database and the workstation database. The architectural schema for this solution is identical to screen scraping, but the software components are slightly different.

**Advantages**
- Mainframe code is untouched.
- Performance gains are possible, since the workstation is equipped with its own database environment, which is tightly coupled to a web server.
- The added capabilities of workstation web server (server toolkits) can be brought to bear.
- There is no (continuous) session overhead on the mainframe.

**Disadvantages**
- "Copy management" becomes an issue. This includes the cost of building and the effort required to maintain replication. Data is only as up-to-date as the last replication. Changes in systems must be
synchronized. A business rule on the mainframe must be reflected in the workstation application, and vice versa.

- Even though there are no sessions, the periodic updates of data must be made across the LAN, and are limited to LAN (rather than mainframe bus) speeds. Apparent performance gains come at a price.
- The addition of web and database servers increases the maintenance burden.
- Workstation reliability becomes an issue. Servers will have to be periodically re-booted to prevent memory leakage.
- System does not support Online Transaction Processing (OLTP). Coding is required on the UNIX or NT server to duplicate the business rules from the mainframe.

Since only rows and columns of data are being transferred between the mainframe and the middle layer systems, development of the web application requires placing business rules on middle layer systems.

Like the screen-scraping solutions, the middle layer server load needs to be distributed and also the middle layer server systems will have to be synchronized with the mainframe system. This alternative may become a high-cost option as the system grows.

Method of Implementation
The following steps (phases) will be followed in migrating applications from one environment to others:

- The starting point for the Legacy Extension project would be a study of the existing application to determine the target architecture that best suits it. The Legacy Extension architecture will be decided based on a number of factors, such as organizational goals, project goals, budget, and time frame.
- In the next step, a prototype is developed with resources such as web server and gateway software.
- After validating the web architecture, the new web applications are developed or the existing application screens will be mapped to HTML pages using tools such as Jacada or CICS web interface.

In this step the web-enabled applications will be tested with the mainframe applications and data. The new outputs, such as reports, file or database updates, and file extracts, should match the existing outputs. Depending upon the application and data volume, tools will be used to find the differences.

Factors Favoring this Approach
The major constraints of the web enabling approach are:

- New technologies require trained people to maintain and administer
- May require additional H/W and S/W or upgrades
Summary of Pros and Cons

<table>
<thead>
<tr>
<th>Sno</th>
<th>Feature</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Technology Enabled Solution</td>
<td>Uses advanced technologies</td>
<td>Requires training for additional resources</td>
</tr>
<tr>
<td>2.</td>
<td>Cost and Schedules</td>
<td>Very low compared to BPR or re-development.</td>
<td>May vary depending upon the Legacy Extension solution</td>
</tr>
<tr>
<td>3.</td>
<td>Additional H/W or S/W or upgrades of existing environment.</td>
<td>Helps the company to keep in pace with emerging technologies.</td>
<td>Additional cost</td>
</tr>
<tr>
<td>5.</td>
<td>Reuse the existing application system and data.</td>
<td>The existing system will not be touched. New Wrappers will be developed.</td>
<td>Duplication of data and applications may occur depending upon the solution.</td>
</tr>
<tr>
<td>6.</td>
<td>% of Success</td>
<td>Very High</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Meet Competition</td>
<td>Using the leading edge technology, organization can reduce costs and better compete in the market</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Aligning with technology needs</td>
<td>Will take organizations to the new frontiers in technology</td>
<td></td>
</tr>
</tbody>
</table>

Legacy Extension Methodology

In Legacy extension methodology, the developer uses leading edge tools either from the existing vendor or from a third-party vendor. The existing environment may require an upgrade. In most of the cases, this approach requires additional middle layer infrastructure such as UNIX or NT application, data, and web servers. In addition, the prototype may sometimes be developed with beta releases of the leading edge S/W.

Compared to other re-engineering methodologies, this methodology requires additional person-power and training. It provides more benefits and the ROI is comparatively very high. The following subsections describe the various phases in legacy extension of applications.

Assessment and Planning:

- Setup a legacy extension group consisting of top management and application experts
- Identify applications for web-enabling
  - Decide tool/vendor for legacy extension
  - Collect inventory of applications that require web-enabling
    * Number of programs
* Number of screens and reports
* Number of files and/or tables
* Number of interface components and relationships
* Type of application and associated complexity
  - Perform high level legacy extension effectiveness
  - Understand the issues involved in web-enabling
- Prepare legacy extension roadmap
  - Estimate project timelines
  - Estimate customer/outside resource requirement
  - Estimate ongoing costs
  - Estimate risk factors
  - Prepare a ball park estimate
- Prepare a high level project plan

**Project Start and Setup:**
- Set up communication infrastructure
- Coordinate space and hardware requirement for the legacy extension team
- Plan H/W and S/W requirements for development and implementation
- Plan and agree on change management procedure
- Discuss and agree on test strategy, test beds, and regions
- Define status reporting mechanism
- Define metrics to measure progress
- Define generic acceptance criteria
- Develop detailed implementation schedule

**Prototyping and Refinement:**
- Identify a sub system for prototyping
- Web-enable the sub system using the decided approach and tools
- Analyze the prototype results
- Refine the legacy extension strategy
- Update project plans and schedules

**Application Web Extension:**
- Prepare legacy extension schedule for sub system(s)
- Enable applications either manually or with automated tools
- Perform the manual touches required and do a review on the manual changes
- Update the approach, if required

**Unit Testing:**
- Prepare and review unit test plan
- Perform testing
- Review test results and take appropriate action

**System Testing:**
- Client to execute acceptance test
- Fix all problems reported
- Review performance
- Sign-off by user
Parallel Runs and Testing:
- Plan parallel testing
- Migrate production data from the existing environment to new environment
- Install the web-enabled system in the new environment
- Run the enabled system in parallel with the existing system
- Compare the test results and take corrective action
- Review performance related issues and take corrective action

Implementation:
- Plan Back-out
- Install the new infrastructure
- Check for security loop holes
- Migrate production data from the existing environment to new environment
- Install the production system

Post Implementation Support
- Provide production support for the enabled system
- Provide orientation to the customer support staff in the new environment and the types of system changes incorporated

Legacy Transformation Approach
In the Legacy Transformation approach, an application is moved from the existing environment to another, without a change in functionality.

If an organization decides to migrate its applications, the first step would be to decide the mode of transformation. The entire process could be performed manually, but automation would be the right choice. Automation makes the process less error prone. It is much faster and more amenable to enhancement. Moreover, in applications having a large number of programs with a number of interdependent factors between the programs, manual re-engineering results in inconsistent results and errors caused by monotony.

Automation of transformation can be effected using compiler technology. This will involve constructing lexical analyzers and parsers using standard tools, analyzing the application programs, and converting the programs to the target platform.

Tools built using compiler technology, are more effective compared to manual effort for the following reasons:

- The scanner and parser generators are a faster method of development, and are very fast in processing applications.
- The inputs to these scanner and parser generator tools are high level specifications. This way, the chances of errors in design are minimized.
A major advantage of using scanner and parser generators is that we can incrementally augment the generated parsers and scanners to handle new constructs and transformations.

The scanner and parser generators are tools that come tested, are a comprehensive tool set and complement each other’s functionality.

The development time that is saved in parser and scanner generation can be utilized in testing and verifying the migration action of the tools.

Using the automated transformation tools, the following application problems can be addressed easily:

- Moving an application from one UNIX (AIX) environment to another UNIX (Solaris) environment.
- Moving a FORTRAN or Pascal application to a C application in the UNIX or NT environment.

**Application Transformation Methodology**

Transformation methodology involves using tools for automating the re-engineering process to the extent possible. An analysis of existing applications and other processes needs to be carried out to determine the best strategy, which minimizes the developer’s perspective.

The following subsections describe the various phases in re-engineering applications using the Transformation approach.

**Assessment and Planning:**
- Set up a re-engineering committee consisting of top management and application experts
- Identify applications for transformation
  - Decide tool/vendor for performing transformation analysis
  - Gather inventory of applications
    * Lines of code and number of programs by language type
    * Number of screens and reports
    * Number of files and/or tables
    * Number of interface components and relationships
    * Type of application and associated complexity
  - Perform high level transformation impact analysis
  - Understand the complexity of transformation
- Prepare transformation roadmap
  - Estimate project timelines
  - Estimate customer/outside resource requirement
  - Estimate ongoing costs
  - Estimate risk factors
  - Prepare a ball park estimate
  - Prepare a high level project plan

**Project Start and Setup:**
- Set up communication infrastructure
- Coordinate space and hardware requirement for the transformation team
- Plan H/W and S/W requirements for development and implementation
- Plan and agree on change management procedure
- Discuss and agree on test strategy, test beds, and regions
- Define status reporting mechanism
- Define metrics to measure progress of transformation
- Define generic acceptance criteria
- Develop detailed implementation schedule

**Transformation Impact Analysis and Development of Cookbook:**
- Perform transformation impact analysis
- Develop a transformation cook book
- Refine existing tools or develop a transformation tool

**Prototyping and Refinement:**
- Identify a subsystem for prototyping
- Migrate the subsystem using the developed methodology and tools defined in the cook book
- Analyze the prototype results
- Refine the transformation strategy and update the cook book
- Update project plans and schedules

**Application Transformation:**
- Prepare transformation schedule for subsystem
- Perform automatic transformation
- Perform the manual code conversion required and do a review on the manual code changes
- Update cookbook, if required

**Unit Testing:**
- Prepare and review unit test plan
- Perform testing
- Review test results and take corrective action

**System Testing:**
- Set up system test environment
- Perform testing
- Review test results and take corrective action

**Acceptance Testing:**
- Client to execute acceptance test
- Fix all problems reported
- Review performance
- Sign-off by user

**Parallel Runs and Testing:**
- Plan parallel testing
- Migrate production data from the existing environment to new environment
- Install the migrated system in the new environment
Run the migrated system in parallel with the existing system
Compare the test results and take corrective action
Review performance related issues and take corrective action

**Implementation:**
- Plan Back-out
- Migrate production data from the existing environment to new environment
- Install the production system

**Post Implementation Support:**
- Provide production support for the Transformation
- Provide orientation to the customer support staff in the new environment and the types of system changes incorporated
Appendix A – Legacy Understanding Tools

Many vendors in the legacy system market represent code and system analysis tools that have been in the market for many years. The Leaders and Visionaries’ products are being led by eBusiness initiatives. This focus provides the necessary knowledge to enable legacy extension and application integration solutions in a way that recognizes the business knowledge already implemented in systems. These tools reduce the risk of missing business logic or misunderstanding application interfaces between legacy systems.

Traditionally, the tools available to enable organizations to better understand system component relationships, or program flow, have been driven by a desire to reduce the overwhelming cost of maintenance. However, the tidal wave of business changes pushed by the impact of e-Business is causing AD organizations to revisit these tools. Extending the current implementation of business rules by creating components, or creating programmatic access in a distributed computing environment, are the new drivers. While there has been an increased interest in software process improvement, the need to understand data and control flow for legacy extension or transition has become the main impetus for these tools.

Tool Capabilities

The legacy understanding tools provides system inventory, program-flow understanding, data and code understanding and code modification capabilities for a wide variety of languages, databases and transaction environments. A number of vendors specialize in language parsers, which assist in understanding multiple computing languages. These parsers are used to present a graphic representation of logic and program flow, as well as calculating quality assurance benchmark measurements in areas such as process completeness and complexity.

Legacy Understanding Product Requirements

- Language parsing/scanning support
- Centralized repository
- Graphical presentation of application artifacts
- Incremental analyze capability
- Flow-charting functions
- Data flow trace
- Forward and backward logic mapping
- Consistent user interface
- User roles
- Detection of source change for re-analysis
- Mapping operational application components
- Impact of change analysis
Tool Evaluation Criteria
Legacy understanding tools should be evaluated for their ability to provide good information about data and control flow, particularly to identify interface definitions. It includes:

- The ability to find potential interface points (e.g., data files DBMS tables, work files CICS messages, screens)
- The ability to enable recognition of system clusters that are bounded by higher-level interfaces
- The capability to perform impact analysis of interface information
- The ability to create interface documentation that can be used to enable proper componentization or wrappers (e.g., data formats)
- Code restructuring/slicing capabilities to add new interfaces or rationalize existing ones.

Legacy Understanding Vendors
This is a representative list of legacy understanding products. Cognizant has tested and used some them. They are included to present the reader with examples of system and program-level understanding tools.

<table>
<thead>
<tr>
<th>SNo</th>
<th>Vendor Name</th>
<th>Tool Name</th>
<th>Web Site</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>SEEC</td>
<td>Mosaic Studio</td>
<td><a href="http://www.seec.com">www.seec.com</a></td>
</tr>
<tr>
<td>5.</td>
<td>Netron</td>
<td>HotRod</td>
<td><a href="http://www.netron.com">www.netron.com</a></td>
</tr>
<tr>
<td>6.</td>
<td>Cast Software</td>
<td>Application Miner</td>
<td><a href="http://www.castsoftware.com">www.castsoftware.com</a></td>
</tr>
<tr>
<td>7.</td>
<td>Relativity Technology</td>
<td>Rescueware</td>
<td><a href="http://www.relativity.com">www.relativity.com</a></td>
</tr>
</tbody>
</table>

The legacy understanding vendors represented in the above table provide detailed code understanding and/or the ability to create bill-of-material relationships among various application artifacts necessary to implement a legacy application. Because of their lineage, most vendors in this space focus on traditional mainframe application environments and languages.

The noted alternative is Cast and its Application Mining Suite of tools. The Application Mining Suite and the various Cast Miner products provide program understanding and artifact relationship for distributed applications. Merant's AssetMiner suite, which combines EnterpriseLink, Revolve and Netron's HotRod product, provides a strong understanding component as the underpinning of this offering. Relativity Technologies RescueWare provides good understanding capability as well, even though it has been positioned as a transformation product.
Software Metrics Vendors: This is a representative list of software metrics products. They are included to present the reader with examples of software metric tools:

- McCabe QA from McCabe and Associates
- Visual Recap from Viasoft
- KnowledgePlan from Artemis Management Systems
- SLIM Metrics from QSM
Appendix B – Legacy Data Profiling Tools

Figuring out the nature of exact data resides in a legacy system is usually the first step when building a data warehouse or an E-business platform. According to data analysts 80% of business data today resides on legacy mainframe systems. Efforts to build Web-enabled procurement, order-processing, and customer-service systems require integrating that data with new e-Commerce or customer relationship management applications and data warehouses.

Before that integration work can begin IT managers must study the legacy data to uncover hidden quality problems and inconsistencies. A bank trying to pull together information from multiple databases to create a single view of its customers, for example, may discover that each database uses a different address format, which can make it difficult to combine the data.

The most significant feature of the Axio technology is that it examines the granular, or detailed, data in a legacy system, rather than simply analyzing the metadata as most data analysis tools do. This product goes through the data column by column, row by row, to figure out the exact details.

EVOKE AXIO
Evoke Axio provides organizations with the ability to gain a thorough understanding of the content, structure, and quality of their corporate systems. Axio’s discovery process reveals hidden data quality issues, inconsistencies, and incompatibilities between data sources and target applications. The result is an accurate description of the source data, which is stored in a central repository and used to accelerate the design and implementation of new systems.

Axio tools analyzes source data through a process called data profiling. Data profiling includes data column profiling, dependency discovery and redundancy analysis and results in a fully normalized view of the source data. This powerful analysis and discovery process reveals key facts about the data including:

- Uniqueness of values in each column
- Minimum and maximum values in each column
- Frequency and distribution of values in each column
- Inferred data type for each column, based on the data type most compatible with the data itself
- Inconsistencies between documented and actual data characteristics
- Functional dependency relationships across data elements
- Data that violates expected relationships across data elements
- Value overlap between data elements suggesting opportunities for data consolidation
This process results in a factual foundation on which data analysts and business professionals can work collaboratively to build high quality business solutions.

Vality Technology Inc.’s Integrit

In addition to Evoke Vality’ Integrity helps to improve the quality of the legacy data. INTEGRITY Real Time™ delivers the unrivaled data re-engineering, which it re-engineers to meet the standards of content and formatting and associates it with relevant existing data. Support for C, COBOL, and Java helps easily integrate INTEGRITY Real Time with high performance e-Business applications. The conditioning process converts free-form transaction data into discrete, structured fields, it identifies invalid values, standardizes spelling, abbreviations, and formatting.
Appendix C – Legacy Extension Tools

Host-to-Web Products

Transaction wrapping is basically the use of the existing 3270 interface as an API. Many products provide this function. Through existing Web technologies, host-to-Web products use communications gateways to interact with existing transactions, emulating an end user at a terminal. Interaction is limited to what was available to an end user at a terminal. This approach can eliminate much of the restructuring activity outlined above. Since this limits the implementations equivalent to that of end-user interactions via screens, it introduces unnecessary overhead and requires multiple transactions to complete new business processes. Gathering data for display this way is acceptable and combining processes on multiple tiers is more difficult.

GUI Extension Product Requirements

The GUI extension products provide a noninvasive process for transforming existing character-based presentation formats into graphic presentations through fat or thin clients. Several graphical user interface (GUI) extension approaches are available and mentioned earlier. The product requirements are divided into development and deployment.

The development requirements are:

- Semi automated screen capture
- Rule/template screen capture
- Semi automated maintenance
- Open interfaces
- Shared development
- Many-to-many screen mapping
- Dynamic value-added graphics
- Key and cursor control

The deployment requirements are:

- Multiple input streams
- High-volume capability
- Multiple server options
- Traffic monitoring
- Logging capability
- Visual tracing
- Integrated to workflow
- Multiple emulator options
Data Extension Tools

This segment identifies products that provide a noninvasive process to accessing legacy data in its current environment, whether relational or non-relational. Data extension products provide an understanding of the implied data models in existing physical files. These physical models can be turned into logical models for further enhancement or extension, or used as the basis to generate different file definition languages. Another set of vendors in the data extension market segment enable the extended use of the actual data for either downstream reporting or for transporting of data for new environments and new integration opportunities.

Data Extension Product Requirements

The product requirements are divided into development and deployment.

The Development requirements are:

- Visual data-mapping capabilities
- Support for relational and non-relational data sources (e.g., DB2, Oracle, Sybase, Informix, VSAM, IMS-DB, CA-IDMS, Adabas, Datacom-DB, PDS, Sequential)
- Interface with existing security packages (ACF2, TopSecret, RACF)
- Data compression and data encryption capabilities
- Access to database through existing transactional systems (CICS, IMS)
- Full Java support
- Data dictionary/repository
- Import source copybook data definitions (e.g., COBOL copybooks)

The Deployment requirements are

- Request logging
- Database journaling/rollback
- Administrative control
- Open Database Connectivity, Java Database Connectivity, Object Linking and Embedding database access
- Multiple network protocol client connections (e.g., TCP/IP, APPC, 3270)
- Automatic data transformation (e.g., ASCII to Extended Binary-Coded Decimal Interchange Code)
Appendix D – Legacy Transformation Tools

Legacy Transformation Tools

The enterprises legacy transformation tools provide business rule identification, code slicing, code modification or transformation from one language to another. These products are generally provided as add-ons to legacy-understanding tools. They may operate on a mainframe or a workstation. Other sophisticated tools can also support language wrapping for creating components out of legacy systems, and provide support for porting legacy business logic to new architectures and languages.

Product Requirements

- Multiple code-slicing strategies
- Mass change
- Data-mapping transformation
- Code/language transformation
- Business rule isolation
- Paragraph/perform range extract
<table>
<thead>
<tr>
<th>Sno</th>
<th>Legacy Environment</th>
<th>Tool Vendor</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unisys</td>
<td>Relativity</td>
<td>Relativity Technologies has had some success helping organizations escape their burning platforms, especially older Unisys platforms. Relativity uses its legacy understanding and re-engineering features to provide Web enablement of mainframe systems.</td>
</tr>
<tr>
<td></td>
<td>SEEC</td>
<td>SWS</td>
<td>SEEC has a relationship with Unisys to aid in the transform of legacy Unisys systems to new platforms. SEEC uses its legacy understanding and re-engineering features to provide Web enablement of mainframe systems.</td>
</tr>
<tr>
<td></td>
<td>Legacy Databases</td>
<td>SWS</td>
<td>SWS has been helpful in transitioning legacy database systems to newer relationship environments, including DB2, Oracle and Informix.</td>
</tr>
<tr>
<td></td>
<td>Non-Mainframe</td>
<td>Transoft</td>
<td>Transoft addresses modernization of non-mainframe legacy platforms.</td>
</tr>
<tr>
<td></td>
<td>DEC, NCR or Unix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Legacy Languages</td>
<td>Trinity</td>
<td>Trinity Millennium Group provides a factory-based offering to enable understanding of a wide range of legacy languages and databases in preparation for a variety of modernization or conversion activities.</td>
<td></td>
</tr>
</tbody>
</table>

All these vendors provide a level of legacy understanding as the basis for their transformation solutions. Although a wide variety of solutions is possible with each vendor in this market, solutions generally are focused on a particular re-engineering activity.
Appendix E – Legacy Client/Server Tools

Legacy Understanding and Transformation Tools

Tools to automate the understanding of existing systems are most often associated with the extensive and aging mainframe application portfolios of the last 20 or 30 years. The volume of code and related application artifacts, the evaporation of application knowledge and the constant maintenance activities associated with these environments has driven product offerings in this market.

As IS organizations review the development activities associated with the second wave of computing, client/server applications, or the early efforts of third-wave, distributed applications, they have identified a whole new category of legacy. While the volume of this category of legacy is not as high, the lack of understanding is. These systems were often developed with few of the disciplines associated with more traditional AD, as proof-of-concepts, "skunk works" or time-sensitive competitive responses, and little effort was placed on version control or configuration management. The price of such undisciplined development must always be paid.

Although system understanding has been commonplace in traditional mainframe legacy environments, the culture of the distributed programming environment has not seen the benefit. The following types of products are available for a number of database environments, Oracle, Sybase and Microsoft SQL. DB2 support is under development. Programming environments supported include PL/SQL, T/SQl, Visual Basic (VB), PowerBuilder (PB), Oracle Forms and Java, as well as a variety of Internet technologies, including HTML/CFML, VBScript/JavaScript/JScript, ColdFusion Script and Perl.

Product suite analyzers can be combined as necessary to support the necessary technology environments in an organization. The integrated data model provides a unified representation of links between.
Appendix F – COBSTAT–COBOL Static Analyzer Tool

There are 109 billion lines of Legacy code in productions. These applications were developed in COBOL, PL/1, and RPG in 80 different dialects. Due to the emergence of the e-Business and CRM applications, the importance of legacy applications and data is becoming more and more important in the Enterprises. It is very difficult to deploy any e-Business or CRM applications without touching the existing legacy systems. Cognizant’s COBSTAT provides the required tools to understand, maintain, integrate, and transform the Legacy applications. COBSTAT provides the required environment so that users will be able to extract the required information from the legacy systems.

COBSTAT Features

The major features of COBSTAT – Cobol Static Analyzer are:

- Application Analysis
  - Program artifact Analysis
  - Artifact Impact Analysis
  - Code Slice
- Program Flow Analysis
  - Procedure Flow Analysis
  - Control Flow Analysis
- Automatic Code Change
  - Code Change for migration projects
- Analysis and Cross Reference Reports
- Data Dictionary using RDB
- Availability on product on UNIX and Linux
- Internet / Web Interface

Application Analysis

COBSTAT Analyses the given COBOL program and provides information on various application artifacts such as variables, files, procedures, programs, copy books, and subroutine calls to other programs. In addition COBSTAT provides information on the Cross Reference between these artifacts.

Artifact Impact Analysis

COBOL Program variables can be studied using the COBSTAT Impact Analysis feature. Analyzer can analyze more than one variable on a single or set of Cobol Programs. The Impact Analyzer finds the propagation of given variable(s) through the source of the program in iterations until there are no more variables for propagation.
Impact Analyzer provides:
- Propagated Variable’s (Synonym’s) Details
- Propagated Variable’s (Synonym’s) Statistics
- Propagated Variable’s (Synonym’s) Cause List
- Propagated Variable’s (Synonym’s) Affect List
- File Variable List of the COBOL Programs

**Code Slice**
Code Slice is the projection of COBOL source code, where a given variable occurs. The Code slicer slices the source lines where the given variable occurs and projects them for the user.

**Data Dictionary Generation**
COBSAT uses Data Dictionary, to store all the application program artifacts and their relationships. The data dictionary contains the following information about application artifacts:

- Program Detail
- Copy Books
- Program Variables
- Data Files
- Synonyms
- Artifact relationships
- Control flows
- Procedure flows.

Using Data Dictionary feature users can analyze the applications in a standalone mode after performing the CPU intensive operations on the applications in the beginning in a batch mode.

**Program Flow Analysis**
The Program Flow Analyzer provides both procedure flow and control flow.

**Procedure Flow**
The Program Flow Analyzer populates the Program Flow Database with list of procedures and sections present in the source program along with the list of Calls and Performs occurring in a Procedure or a Section.

**Control Flow**
The Program Flow Analyzer populates the Program Flow Database with list of Control Statements and Sections present in the source program along with the list of Calls and Performs occurring in a Section.

**Automatic Code Changer**
COBSAT helps migration projects, to migrate a given application from a given to environment to a new environment. In addition it can be used in application projects, where field expansion is required such as HIPAA,
Decimalization, and Bar Code Expansion. The impact analyzer provides information and reports, on the impact of the migration or field expansion in a given program. The Automatic Code Changer produces Modified Source Code if changes are made to the program source code during the run of COBSAT.

Salient Features of COBSTAT
The next version provides the following additional features:
- MS Windows version with SQL RDB
- Control Flow Analysis
- Variable Flow Analysis
- Business Rules Association
- Field Expansion of variables

Impact Analysis Reports
COBSAT provides the following impact analysis reports:
- Synonyms Report
- Program Statistics Report
- Variable Statistics Report
- Cause List Report
- Affect List Report
- File Variable List Report

Cross Reference Reports
The various Cross Reference Reports are
- CR Calling X Called Program Report vv
- CR Program X Copybook Report vv
- CR Program X File Report vv
- CR Program X Call Report
- CR Program X Entry Report
Appendix G – Cocktail Compiler Tool set

Cognizant used Cocktail Compiler tool set to build the COBSAT COBOL Application code static analyzer. This Compiler tool sets support around 25 different dialects of COBOL Grammars.

Cocktail toolset comprises of tools, which assist in various phases of a language processing tool development. These cocktail tools (toolset) help developing tools, which would do the actual language processing. The toolset consists of tools for every component of a typical language-processing tool such as scanning, parsing, attribute evaluation, attribute transformation, and code generation.

The tools comprising the Cocktail toolset themselves accept input in a high level language and generate the code in 'C' programming language, thus hiding the inner complexity of tool generation from the user. The tool generators task is reduced to specifying the behavior of the tool in the high level language understood by the tool. The tools comprising the toolset are complete and can be used in isolation. Powerful language processing tools can be developed using the Cocktail toolset.

Generating Language Processing Tools
A language Processing Tool usually consists of several modules where every module handles a certain task. In principle the LPT model works as follows: A scanner and a parser read the source, check the concrete syntax, and construct an abstract syntax tree. They may perform several normalizations, simplifications, or transformations in order to keep the abstract syntax relatively simple. Semantic Analysis is performed on the abstract syntax tree. Optionally attributes for code generation may be computed. Afterwards the abstract syntax tree is transformed (into an intermediate representation). The latter is the input of the code generator, which finally produces the output code.

Scanning and Parsing
This phase involves reading the input source code, breaking it into tokens and checking the syntax of the constructs formed by these tokens against the grammar of the language. This phase generates the parse tree for the input code. The Cocktail tools REX helps in generation of scanners. The input of REX is specified in form of regular expressions that constitute the valid tokens. The Cocktail tool LARK, helps in generation of parsers. These parsers parse the LALR(1) and LR(1) grammars. It supports named attribution and backtracking.

Semantic Analysis and Transformation
This phase involves validating the tree for semantic correctness. The necessary transformation can be made to the tree for achieving the functionality of the language-processing tool. The output of this phase is the transformed tree.
The Cocktail tool AST helps in specifying the abstract syntax of the tree. This helps in mapping the concern syntax of the grammar to the abstract format. The various attributes can be specified. The various attribute computations necessary for semantic analysis can be achieved easily through this tool. The Cocktail tool PUMA, helps to generate the tree transformation routines. Tree transformation becomes very easy as PUMA works on pattern matching and recursive routines. PUMA allows attribute manipulation and transformation. PUMA cooperates with AST and can use the abstract tree generated by AST.

**Code Retrieval**

The last phase of the language processing tools involves getting the transformed code back. This phase requires mapping the abstract tree back to the language of the input code. As lot of formatting information is lost in generation of the tree, it has to be specifically stored as attributes. The PUMA routine can be used to traverse the tree recursively and specific calls can be made to hard coded routines for mapping the tree to the native syntax of the language.

**COBSTAT Tool Set**

Using the Cocktail tools such as REX, LARK, AST, and PUMA, Cognizant built the various features of the COBSAT. Using Cocktail tools one can build Compiler based tools by giving higher level specifications. In other words, one can build the Compiler tools without writing much hand written C code, which is prone to errors. The major advantages of using Cocktail tools in building the COBSAT are:

- Supports a number of COBOL dialects
- Stable and expandable
- Nearly error free