
Legacy Transformation ROI and Case Studies

Business Rule Extraction and Transformation of COBOL Applications

Version 2.2
www.softwaremining.com





Table Of Contents

- INTRODUCTION 3**
- MODERNIZATION KEY DRIVERS 4**
- ROI: THE LONG TERM PICTURE..... 5**
- SOFTWAREMINING: MODERNIZATION SOLUTIONS 6**
 - SOFTWAREMINING Services: Outsourcing..... 6
- ALTERNATIVE STRATEGIES 7**
 - Replacement with Packaged Application:..... 7
 - Manual Rewrite:..... 7
 - Outsourcing the entire application:..... 7
 - Quick Fix Solutions: Addition of more Layers:..... 7
- CASE STUDIES 8**
 - Case Study I – Addressing Internal Development Issues..... 8
 - Case Study II. Obsolete Hardware; (Telecoms Company) 9
 - Case Study III – Software Company (Fujitsu)..... 9
 - Case Study IV – Reducing Risk; (ACS International). 9
 - Case Study V - Integration..... 10



Introduction

Most IT systems are subject to continuous change due to business and technological advances. 'Change' is at the heart of running successful businesses. A recent statistic quoted that 50% of ABB's income are from products that are less than 3 years old, 50% of Chrysler's income is not from the Manufacture of Cars! Stagnant companies tend to die!

**Change is at the heart of running successful businesses
Stagnant companies tend to die!**

At a certain stage in the life of a legacy business application, the cost of implementation changes can become prohibitively high because the original system was not designed to handle the new requirements. For example many legacy applications do not use a Relational Database (such as ORACLE), or the new Software Development tools and languages (Java or C#) that offer shorter development lifecycles, superior communications and better maintenance. Adopting a more 'open' technology generally improves the productivity and longevity of the applications, as well as opening the door to pools of new programming resources.

Legacy Challenge:

- **How many people in your organisation know the nature and extent of your software assets?**
- **The average age of Legacy developers is increasing. What is the your long term strategy for application maintenance? What happens when the developer team retires, or if due to skills shortages they are headhunted by competitors?**
- **How much is the implementation of business-driven changes costing your organisation?**
- **How much can the modernisation of the legacy applications reduce costs and risks of future enhancements?**
- **Most goods, such as cars, undergo regular major maintenance services. When is the next major maintenance of your legacy software?**

SOFTWAREMINING Tools and Services address the future well being of the legacy code via Complete Modernization Solutions - Extracting the Business Rules and moving it into new languages and architecture.

Apart from business-driven changes, another set of challenges are being continuously forced on IT departments. These new changes reflect the business opportunities offered by new technology. For example, over the past few years most IT departments have had to shoehorn the following changes into their existing legacy applications:

- GUI interfaces
- Y2K
- Euro
- B2B integration
- Hardware obsolescence
- Web-enabling

Whilst the implementation cost of each of the above layers remains small in comparison to a complete over-haul involved in modernization effort – the total costs remain comparative.

The only certainty is that maintaining your competitive and technological edge will demand continued enhancement and development of your IT Systems.

There are a multitude of indicators pointing to an eventual exponential growth in the maintenance and enhancement costs for legacy applications. These include

- Increased dependency of the system on the different layers for handling XML, GUI interfaces and Service based architecture
- Use of a diverse range of proprietary technology and components – leading to a more complex system and, in time, integration issues
- Less resilience to turn-over of key staff

These problems are best addressed via Modernization of the application.

SOFTWAREMINING specializes in tools and services to ease the Modernization effort.

Modernization Key Drivers

Some Specific Deficiencies inherent in Legacy Applications are seen to be:

- Ever shrinking pool of legacy programming resources
- High cost of maintenance of legacy hardware
- Hardware Obsolescence - as in HP3000 mainframe
- Proprietary system: UNISYS Databases, NATURAL Language, COOL:Gen Code, IDMS.
- Web-enabling
- Integration, Connectivity, B2B exchanges, trading hubs
- Normalization of Applications

This is where legacy applications restrict or prevent the agility so desperately needed in today's business world. In a competitive environment those businesses with cleaner code, better-documented systems and more flexible technology/architecture/framework can adapt and re-adapt much faster than those who need to apply changes to multiple layers of legacy code.

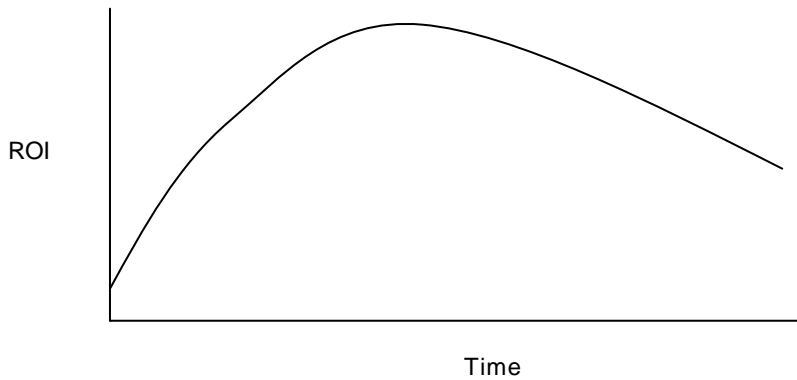
Modernization can hence achieve the following benefits

- Reduction of IT hardware and software costs through the standardization of IT platforms (replacing mainframes with cheaper servers)
- Corresponding reduction in head count
- Rationalization of existing applications (reduced duplications)
- Improved operational efficiency and effectiveness via the sharing of IT platforms
- Integration of IT systems with clients & suppliers - leading to automation of accounts payable, streaming of supply chain management, reduced inventory costs, etc.
- Removal of expensive, 'specialist' systems with high support costs
- The production of 'cleaner' applications (removal of dead code) in a scalable environment - one result being increased throughput and thus greater operational efficiency

ROI: The long term picture

Business applications invariably require change – the addition of new processes, integration with other applications, software/hardware upgrades and etc.

For such applications - the ROI can be represented as:



ROI for investment in further enhancements of legacy applications

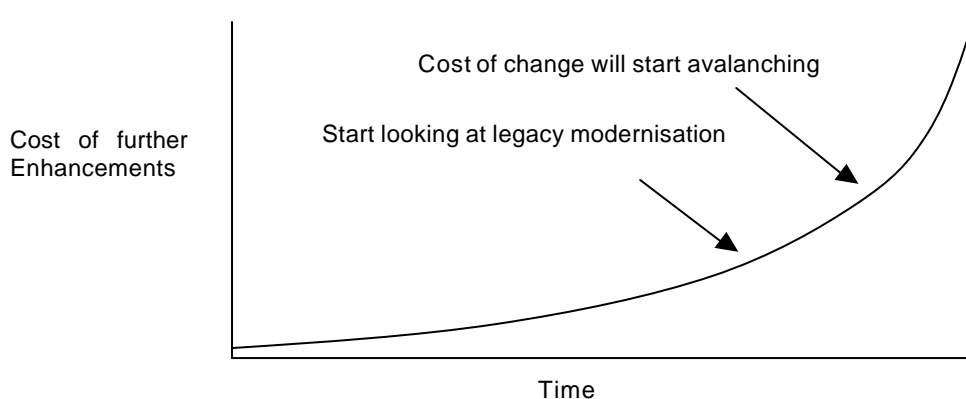
The graph shows that after the initial development stage the cost of maintenance and enhancement of an application is low, yielding increasing Return on Investment. However, at a certain stage, the enhancement cost will begin to grow exponentially – leading to a reduced ROI for subsequent changes.

***This model does not show a new concept – maintenance costs do increase with time.
e.g. old cars, houses, or hardware***

Old hardware requires regular major maintenance services and even then, at some stage, it will cost less to replace. Similarly, Legacy Software needs Modernization to reduce its running costs – and extend its usefulness.

The maintenance and enhancement costs increases for a variety of reasons. For example, the dependency an increasing number of proprietary components can lead to large impacts for even small changes, or the turnover of key legacy staff has a major impact on the maintenance of the system.

The loss of knowledge and impact of changes can only have one possible outcome: Increased Cost of Enhancements. The cost of change over time is represented in the following graph:



Cost of Enhancement grows over time due to loss of knowledge and greater impact of changes

SOFTWAREMINING: Modernization Solutions

SOFTWAREMINING's Modernization Solutions are based on our Automatic Transformation Toolkit: CORECT. CORECT has a proven track record and offers the following features:

- **Documentation of Legacy Systems** – What are the existing legacy assets in the company?
- **Business Rule Extraction** – How does the system achieve business processes?
- **Architecture Driven Modernization**: clean up code, architecture discovery, re-host or transform
- **Catering for the future direction**: Support for OMG's Model Driven Architecture (MDA), UML, XMI and the new OMG Architecture Driven Modernization.

*The primary objective of SOFTWAREMINING is to produce **legible and maintainable code** - allowing developers to easily enhance and extend the application.*

SOFTWAREMINING's Modernization tool (CORECT) offers an economic means for the modernization of legacy applications. The toolset works in 2 modes:

- **AUTO-Cleanup (Default)**: Using Artificial Intelligence and heuristics techniques, the auto-cleanup process will identify and remove dead code / variables and introduces an Object Oriented Framework into the generated code.

The biggest advantage of working in Auto-Cleanup mode is **rapid delivery time**. With a Transformation and Compilation rate of up-to 10,000 Line Of Code / Man day – this is some 100 times faster than the productivity of developers manually rewriting the application.

Typically the transformation of a medium complexity application of 1 Million Lines of Code can be achieved using 5 people in 3 or 4 months.

- **Business Rule Extraction (BRE)**: When a major architectural renovation of the system is required – then the Business Rule Extraction module facilitates the extraction of business rules and processes from the application.

Programs are typically involved in performing a variety of operations – business process, screen handling, transaction handling, database read/write/update and etc. The BRE module allows isolation of code belonging to different categories (e.g. identify and display only the business process code), as well as identification of code working on different records (e.g. identify all the business-process code and transaction handling code which operate on 'ORDER' and 'ORDER-LINE' records).

SOFTWAREMINING Services: Outsourcing

The Modernization Service performed by SOFTWAREMINING's offshore facilities offers the following advantages to our clients:

- **Faster turn-around**: no training or staff learning-curves
- **High Quality of Work**: Modernization done by domain experts
- **Cost Effective**: on-par with costs of doing the project internally
- **Retraining**: Gives client's staff time to retrain in new languages (Java, C# or Oracle), and become familiarized with the new system

Alternative Strategies

Replacement with Packaged Application:

Packaged applications, being non-bespoke, inevitably carry large potential limitations. It is estimated that 35% of new ERP implementations are cancelled, and only 10% of them meet the entire client requirement.

Manual Rewrite:

Manual rewrite has major limitations including:

- Prohibitively high cost
- 70% risk of failure
- Lengthy timescales - often meaning that requirements change in the meantime!

Outsourcing the entire application:

Outsourcing changes the ownership of the problem. The outsourcing companies are subject to the same deterioration of knowledge and have to solve the same problems. However, they may be in a better position to undertake the modernization initiatives.

Quick Fix Solutions: Addition of more Layers:

Some problems can be addressed via additional layers to the system:

- Character screens to GUI interfaces Upgrades – Screen Scrappers
- CORBA Communication
- Web-enabling
- Service based architecture.

Such issues have typically been addressed in this way but the introduction of numerous layers leads to increasingly difficulty with the implementation of *business changes*. Modifications have to be made to each different layer. It is inevitable that at some stage the effort in maintenance of the layers will overtake that of the business. It is time to go back to the drawing board!

Case Studies

Case Study I – Addressing Internal Development Issues.

A leading multimedia company has a host of COBOL applications essential for the day-to-day running of business.

Issues:

The IT department faces the following problems:

- The average age of the programming staff is increasing
- The application system is not fully documented
- Not all the business processes are automated, the programming staff are currently manually starting, linking and completing some processes
- Some hardware leases are close to expiry
- Further investment in the development of additional software in the hardware platform is viewed expensive considering the lifespan of the existing hardware platform

Objectives:

The company was looking for a solution to address the following:

- Re-documentation of the system
- Allow the Legacy programming staff to continue the maintenance of the new system with a limited amount of re-training
- Allow a gradual influx of new-programmers to take over the development of new processes, familiarize with the existing system i.e. to reduce the dependency on the Legacy programming staff
- The future investment for enhancements to utilize and benefit from more up to date technology

Solution:

SOFTWAREMINING solutions addresses these issues by:

- **Generating Documentation, Extracting Business Rules**
- **Translation into Java:**
 - **Legacy programmers:** After limited retraining – the Legacy programming staff can still trace their own work in the new language. This contrasts significantly with having Legacy Developers writing a new system in Java from scratch.
 - **New Java /Oracle programming staff:** Involved in administration of new technologies: ORACLE, WebSphere, iPlanet, as well as development of new modules and gradual familiarization with the modernized system.
- **Proprietary Platform;** With Java and Oracle as the new development platforms, the dependency on old proprietary platforms is removed, Further development can benefit from the offerings of the latest technologies and further investment in the application is no longer viewed as wasted resources.

Case Study II. Obsolete Hardware; (Telecom Company)

A South American Telecom company has in excess of 2 million lines of COBOL code. The COBOL code runs on UNISYS platform with utilizing UNISYS Databases. The initial requirement is to move the application onto an open platform. The options are Java/Oracle or MicroFocus COBOL/Oracle. The target platform is Sun Solaris.

The Primary Business Criteria is Performance and Application Integrity.

During the Proof-Of-Concept study, SOFTWAREMINING analyzed the original source code and produced and generated MicroFocus COBOL and Java code – to work with a 'common' Oracle Database.

After the evaluation of the generated code/application – the following decisions were reached:

- The Java code had significant better future upgrade paths
- Performance of Java/Oracle was significantly better than COBOL/Unisys DB
- The high quality of generated code enabled the existing COBOL programmers to re-train and continue the maintenance of the system in co-operation with a new Java team

Case Study III – Software Company (Fujitsu).

FUJITSU has developed their Logistic System over the last 20 years. The system has been written for Fujitsu mainframes (COBOL) and has a substantial client base in Japan.

The Modernization Strategy aimed to

- Revitalize products sales and upgrade
- Open the product to a wider PC market
- Better position FUJITSU to compete in today's competitive marketplace

SOFTWAREMINING's Tools were the only toolset to pass all the extensive POC tests – and is being used to successfully migrate this system.

Case Study IV – Reducing Risk; (ACS International).

A major Indian Outsourcing organisation has won a contract to re-write the legacy system for an American client.

Whilst the cost associated with re-analysis and development is low, the delivery of the project in a short space of time requires too many 'bodies'. Merely the initial manual Proof-of-Concept project involves a team of 30 analyst and development staff. This model was not scalable when moving from POC to the main body of code, or handling migration projects from other clients.

Additionally, the risk involved in delivering large projects - with more than 100 staff - is high.

SOFTWAREMINING's tools were utilized initially to do business rule extraction from the code – i.e. reducing the analysis stage, followed by delivery of Java Business Rules.

ROI using SOFTWAREMINING's tools:

- Addresses the scalability: Reduces the project staffing level
- Reduces risks associated in large project (staffing reduced by a factor of 10).
- Reduces delivery times,

Case Study V - Integration

A large insurance company has a host of IT systems for Treaty Management, Claims Administration, Policy Administration, Account Management, Underwriting, Financial Management and Payment. This functionality is provided via a series of custom made applications written on different platforms.

The main focus in integration is communication across different parts of a multi-platform system. However, provision of external channels to suppliers and clients (EDI and Extranet) is also needed. All the new channels are expected to re-use the existing internal components. Therefore all the components are to standardise on use of XML as a means of communication, i.e. all system interfaces, internal, external and EDI components to communicate with each other via a XML.

Software Mining's CORECT Tool generates code that can meet the XML level integration required here. The generated code utilises a flexible architecture with such requirements in mind. The framework layer may be adopted to work with XML messaging without any need to change a single line of business code.

Critical System:

2000 Arizona State County, 4 Million Lines of Legacy Code, development staff diminished from 6 to 1, outdated documentation, high hardware maintenance costs.

Solution: Modernize