ENTERPRISE APPLICATION MODERNIZATION - UNLOCKING LATENT VALUE

INTRODUCTION

Corporations have over the years, deployed a diverse mix of software and hardware applications to gain competitive advantage. Rapid advancements in IT, combined with evolving business needs, have resulted in contrasting IT environments across enterprises.

At one end of the spectrum are open architecture applications that leverage on the potential of the of Internet, while the other end comprises traditional, close-ended, legacy software. corporate data still resides on legacy Some market research estimates indicate that more than 70% of systems. Hence the successful management and re-deployment of legacy systems to meet tomorrow's business needs is the major challenge today.

This white paper will help organizations understand the issues involved in effective management of existing legacy systems.

LEGACY APPLICATIONS

DEFINITION

A legacy system typically consists of large applications that access voluminous data stored in legacy database management systems running on mainframes or mid-range platforms.
These systems made economic sense when they were developed. The functionality of these systems was unquestionable at the time of installation. However, as technology and business needs evolved they have become complex and uneconomical to maintain.
As the enterprise has invested a considerable amount of time and money in these systems, these investments cannot simply be written off.

PROBLEMS ASSOCIATED WITH LEGACY APPLICATIONS

In their current state, most legacy applications have several challenges associated with their functioning and maintenance. A few of the typical challenges are enumerated below:
. Legacy systems are built for internal, enterprise-wide usage, while today's business demands that they be exposed to new, external entities. The focus was almost always on internal business logic.
. These applications are inflexible. They are not modular thus segregation of presentation, business and database logic consumes critical resources.
. The lack of documentation and skilled manpower make any modification an ad-hoc process and not a holistic one. This can lead to crashes and breakdowns in unpredictable parts of the system.

Efforts to address these challenges have been piecemeal, and have had limited impact. The combination of new systems and retrofitted older ones have ompounded the problem. Adoption of new technology and languages has often been only for technology's sake. Finally, the need to deliver application functionality via new channels like mobile devices, with differing transaction approaches add to the problems of successful legacy modernization.

BENEFITS OF LEGACY APPLICATIONS

Organizations continue to use legacy applications on account of various reasons. Some of the benefits perceived by organizations are:

. Legacy systems were developed for, and still run, mission-critical applications.
. A large number of users utilize the system. They are very familiar with the functionalities of the applications, including look and feel. They have also gained a complete understanding of the strengths and limitations of the system.
. The underlying hardware and software of such legacy systems is time-tested and very reliable. The applications themselves have evolved over a few decades and behave very predictably.

These factors contribute to the continued usage of legacy systems. However, effective modernization of these legacy systems will ensure that these benefits can be amplified at minimal expense.

CHANGING BUSINESS REQUIREMENTS AND LEGACY APPLICATIONS

Economic and political conditions over the last few years have resulted in several emerging challenges for technology organizations.

. Time to market is going down. Organizations are moving from 18-month project cycles to 6month project cycles.
. New products and services are being introduced in rapid succession.
. With globalization and deregulation, the need for flexible systems that can synchronize with rapid business shifts has become crucial.
. Organizations are mapping cost controls to appropriate service expectations.

Such dynamic considerations have made it imperative for organizations to assess the financial viability of their IT portfolio, so that they can leverage the advantages of new-age languages and optimize returns on investment on existing applications as well.
Business leaders must consider following strategic issues when evolving beyond legacy systems:

Total Cost of Ownership -

Typically, the Total Cost of Ownership (TCO) of keeping a legacy system running can be very high as compared to the cost of running a more up-to-date system.
The TCO of a system includes components like operations (hardware, system software), production support, and application maintenance. The lines of code, quality of documentation, and the way the application is structured directly influences costs of the system.
Industry experience suggests that maintenance costs drop by as much as a factor of 3 after a legacy system is transformed. This is indeed possible if the newer system is better structured, better documented and has optimized code.

Productivity -

A legacy system typically owes its stability, scalability and reliability to the underlying mainframe platforms on which it is deployed. Any approach to modernizing the legacy system should recognize this and develop a solution accordingly.
Modernizing legacy does not imply migrating away from the mainframe platform in its current manifestation, but optimizing the existing system for enhanced performance.

Flexibility -

The technologies used in a legacy application often do not integrate well with newer technology application components that have been subsequently developed. But the main flexibility loss arises from the fact that the applications are monolithic -- unlike the more recent multi-tiered architectures where the presentation and business logic are separated. Multi-tiered architectures allow for greater flexibility and changes can be effected quickly.
Architectural rigidity is one of the primary reasons that several organizations prefer to re-architect the legacy application, even while retaining the underlying platform and language.
A transformed application makes for a multi-tiered, adaptable system, allowing easy integration of newer technology.

Knowledge Availability -

Programmers adept at COBOL, PL/1, Assembler and several other legacy languages are a vanishing tribe. These programming languages are no longer taught in computer science courses at schools and training institutes – hence, without considerable re-training, it is difficult to create these skills in-house.
The same problem holds true for database technologies used. In the past, hierarchical and network databases were very commonly used, whereas recent applications work with relational databases.

Last but not least, documentation in respect of the application's functionalities is almost always inadequate, and only a few people possess complete knowledge of what the application does.

Extinct Vendors -

In many reported cases, the company that originally developed the application is no longer in business. That leaves their customers in a very precarious position because most often the language used to develop the system is already obsolete and no longer supported. Additionally, the system has usually been heavily customized, and there is no proper documentation maintained. This is also the main bottleneck to implementing enhancement and changes to the application.
Hence, whenever such a system has to be taken over for maintenance, it requires a high learning curve. This period can vary from 2 months to 6 months depending on the complexity of the system. Only after getting familiar with the system can a third-party be able to carry out an effective maintenance job.

Alignment with Business Goals -

Some CIOs certainly do wonder whether it is worthwhile to spend on maintaining and upgrading a legacy system. In reality, such outlay can produce a healthy return on investment should not be considered as mere running costs.
There are three distinct types of maintenance costs: preventive (e.g. Y2K, Euro), adaptive and breakdown. Preventive and breakdown maintenance expenses are necessary to keep the system running, so the costs allocated to these can be said to be running costs.
Adaptive maintenance commonly refers to enhancements or upgrading. This maintenance, though piecemeal, does improve the functionality, accessibility, and provides good business value. Unfortunately, most enhancement requests take a back seat as most budget allocations being consumed by preventive and breakdown maintenance.
Improved returns on investment can be obtained only by undertaking a sizable upgrade, and that too when the business needs it the most.
Proper planning and Return On Investment (ROI) analysis should be done for legacy upgrade to know whether value accrues from increased returns or reduced TCO (maintenance, infrastructure and operational costs).

ALTERNATIVE SOLUTIONS

Organizations moving away from legacy systems must adopt a financial viable solution that meets strategic business needs. There are various options available to the CIO when metamorphosing from legacy systems to more contemporary platforms.

Functional Extension -

Functional Extension is useful when the legacy application possesses adequate business logic, but needs additional functionality.
Functional Extension refers to closing the functional gaps in the legacy application by reengineering the existing application or by integrating it with other application.

Technical Extension -

Technical extension is useful when existing legacy applications have high operational costs and there is a strong need to share the business capabilities with partners/suppliers. One of the key drivers for technical extension is a need to web enable the legacy application.
Technical extension covers activities like:
. Code cleansing / optimizing
. Componentization
. Development of wrappers
. Legacy Integration

In both the functional and technical extension, the processes and business rules are preserved while critical components of the application are converted and adapted.

Migration -

Migration becomes an important modernization option when the legacy application has adequate business rules, but requires higher scalability and interoperability.
This option is also useful, when it is difficult to separate logic from persistent data and presentation layers.
. Selection of targeted programming language/platform/database
. Code migration
. Database migration
. Deployment migration

Replacement -

Replacing the existing legacy application with a generic off-the-shelf product or rewriting it under a new programming environment is another option.
Replacement would accrue benefits similar to re-engineering and is vulnerable to similar disadvantages. There is also the danger of overlooking important business rules that constitute the heart of the legacy application.

SELECTING SUITABLE SOLUTION

The selection of any of these four options would be based on an extensive analysis of the application portfolio around various application parameters, some of which include:
. Functional suitability
. Availability of various features
. Scalability
. Interoperability
. Maintainability
. Reliability
. Availability of standard solutions (OTS Products)
. Ease of use
. Level of documentation available
. Accessibility
. Support available from platform/technology vendor
. Applicability of Enterprise Architecture policies and standards

Portfolio analysis around these parameters will help analyze the applications based on functional gaps and technical gaps within the applications. Once the portfolio analysis has identified the functional and technical gaps, each application can be placed in one of the 9 blocks, shown in the following analysis grid. This will help in identifying a suitable modernization strategy for the application.
Portfolio analysis is the most critical aspect of the overall enterprise application modernization exercise and hence there should be a tool-based approach that would remove, to a great degree, subjectivity introduced by a pure manual approach.

LEGACY EXTENSION (FUNCTIONAL AND TECHNICAL)

WHAT IS LEGACY EXTENSION?

Legacy Extension bridges the gap between legacy and strategic architectures. It augments noninvasive integration and other project options. Legacy extension is cost-effective, time-efficient and risk adverse. The extension process consists of understanding and documenting the existing system; decomposing the application into data, presentation and processing logic; creating and extracting reusable components; and if desired, converting the legacy code into Web compatible languages.

ADVANTAGES OF LEGACY EXTENSION

Extending a legacy system offers organizations a number of distinct advantages including:
1. Up to 40% reduction in maintenance costs, with enhanced understanding of the functionality of your applications. Optimized cost of ownership of transformed system and reduced overall costs (inclusive of new resources, training and maintenance).
2. Leveraging current business processes and modern technology.
3. Improved access to the system through re-deployment and re-orientation of existing hardware and software resources. Anytime, anywhere, secured access to users and customers. Easy access to users over the Internet since no additional hardware or software is required to access.

the application. User-friendly interface that requires minimal training / re-training.
4. Shifts dependence of maintenance activities from few individuals to transparent processes and tools. Ease of maintenance from a Programming / Maintenance group perspective.
5. Comprehensive documentation of system with complete knowledge of processes.
6. Ease in deployment and enhancement of functionality.

METHODS OF LEGACY EXTENSION

Legacy systems typically consist of billions of lines of code in myriad traditional languages. The extension process involves scanning code, extracting business logic, removing dead code and arranging modules into logical components. Skilled programmers can execute these activities manually. However due to various time, cost and risk implications of manual intervention, tool-based extension is a faster, easier and more cost-effective option.

TOOL-BASED APPROACH TO EXTENSION

The demand for rapid application development, along with significant advances in software development automation, has resulted in the creation of tools that automate and aid in the process of legacy extension. In legacy systems, a single program performs multiple functions, or multiple programs may perform a given function. Understanding all operations executed by a function is a difficult task in terms of magnitude, effort and complexity. Several programs may have to be analyzed to completely understand a single function. This method is time consuming and prone to error.

ADVANTAGES OF USING A TOOL

Tool-based extensions can prove to be advantageous in:
1. Extraction of business logic - A tool can extract the business logic related to the functionality, from all the programs and make the entire functionality available in the form of a business rule repository. With the automation of functional analysis, the developer can spend more time in optimization and componentization of
the code.
2. Extraction at system and functional levels - A tool can extract business logic at a system level as well as functional level. Deploying a tool ensures that the complete business knowledge is extracted from the system, while providing an accurate picture of the application(s) functionality.
3. Pictorial depiction of system flow - A tool can also provide a pictorial representation of the system flow, and highlight various modules in the program. This offers the developer a better understanding of the system. Tools can also be used for data migration efforts, whereby it is possible to model data for the target system. This is very useful in cases like VSAM to RDBMS conversion.

Typically, a tool-based approach to legacy extension involves the following steps:

Baselining the Inventory -
1. Tool captures a module-wise inventory.
2. Missing routines, programs etc are reported. For example, program A invokes another program B, and program B does not figure in the program inventory. Program B can then be imported into the tool inventory.
3. The cycle goes on till the inventory is complete.
4. Redundant programs, i.e. the programs that are not referenced by any other programs are identified and ignored.

Planning and Scheduling -
1. Imported programs are analyzed for their complexity. Different tools use different algorithms for determining the complexity.
2. The complexity analysis helps in effort estimation for extension of the programs and further planning & scheduling of necessary activities.

Generating the Process Flow -
1. The tool generates a process flow for a transaction.
2. It highlights the cross-reference and interdependence between programs, batch jobs, modules, etc.
3. The visual representation provides a better understanding of the system at macro and micro levels.

Data Modeling -
1. The tool generates an “as is” data model of the current system.
2. This model can be further normalized and optimized to suit the client’s requirements.
3. This data model can be exported for direct utilization by standard tools such as Rational Rose, ERWIN, etc, to create the target database.
4. The model can also be used to create a DDL for the target database. This feature adds more value when transforming from VSAM datasets to RDBMS.
5. Dependencies and relationships between the various entities can be modeled using graphical interfaces.
6. In most cases, the back-end can remain unchanged.

Knowledge Mining and Extension -
1. Complex rules are split into independent atomic rules. The extracted rules are reviewed and validated against the code and the current functionality. Redundant code is weeded out.
2. “Use Cases” are designed and appropriate business rules are associated with them, thereby building up the components that get translated into software in the target language. A component can consist of more than one function. The design of the components is dependent on the target architecture and infrastructure.

Deployment -
1. The re-architected application is exposed to internal users for testing its functionality.
2. The software generated is implemented on the target platform.

PATNI APPROACH

VALUE-ADDED MAINTENANCE

Patni believes that the best way to service a customer’s need is to imbibe the processes prevalent at the client’s site and blend them with Patni’ s development tools, processes and methodologies.
This approach enables Patni to provide the “best-fit service processes” that add value to the client’s IT operations.
Patni has a ‘Center of Excellence’ for Legacy Modernization. The focus of this group is to:
1. Provide in-house consulting and set benchmarks for a range of Legacy technologies.
2. Identify ‘value-add’ tools, processes and methodologies, and facilitate their usage at client sites.
3. Provide “proof of concept” and formulate solutions in e-Business, Legacy modernization and Application Management.
4. Provide cost-effective solution transfer services to Delivery Units, using a judicious mix of onsite-and offshore-based highly skilled IT professionals.
The Legacy Modernization Center of Excellence possesses expertise in executing projects on a variety of legacy platforms such as IBM mainframe and AS/400, Vax/VMS, HP 3000/MPE.

NON-INVASIVE

Patni believes that any extension of legacy systems should be as “non-invasive” as possible. As described earlier, Re-facing, Re-engineering and Replacement are the three strategies of migrating from legacy to newer platforms. These range from the “cosmetic” to the “highly invasive” methods used by vendors of specific tools and technologies.

SCOPE CUSTOMIZATION

Based on our extensive consultancy experience, Patni scopes out a cost-benefit classification. On the basis of their study, our analysts categorize applications into one of the four categories:

Upgrade / Replaced -
Application that do deliver strategically significant functionality, but have a high cost of retention, have to be retained. However, they are candidates for cost reduction through technology upgrades or through exploitation of other systems. If exploitation of Quadrant 4 (Export) systems makes it possible to replace these systems, these applications will effectively move into Quadrant 1 (Retire)

Retired -
Applications that do not deliver any strategically significant functionality, but have a high cost of retention, are poor value for money. system that have been semiretired, or are used for historical data reference only, would be included in this category.

Retain -

Applications that do not deliver any strategically significant functionality, but have a correspondingly low cost of retention, are best retained on an as is basis. There's not much to be gained from retiring them, as they have a low cost of retention -- nor is there much to be gained from any further investment of time or effort. If exploitation of Quadrant 4 (Exploit) system makes it possible to replace these systems, these applications will effectively move into Quadrant 1 (retire).

Maximize Utilization -
Applications thet do deliver strategically significant functionality, and also have a low cost retention, appearto offer good "Valu for money " and should be utilised as extensively as possible. Exploitation could result in making other (Quadrant 2 upgrade/ replace) and Quadrant 3 (Retain) systems redundant, thus effectively moving them to Quadrant 1 (Retire). High Low Strategic value High

LEGACY APPLICATION EXTENSION PROCESS

Steps:
1. Legacy Understanding: Documenting existing system.
2. System Decomposition: Application is broken into data, presentation and processing logic.
3. Componentization: Create and extract reusable components.
4. Extension: Convert legacy code into Web compatible languages.
Any legacy extension will require the right tools and the right approach. Patni has strategic alliances with some of the leading “legacy modernization” and “Web-enabling” tool providers in the industry. Rich experience, customer-orientation, state-of-the-art development tools, processes and methodologies enable Patni to provide the "best-fit service processes" that add value to the client's IT operations.

CONCLUSION

1. More than 70% of corporate data still resides on legacy systems.
2. Large corporations have invested considerable resources on these systems. This investment cannot be written off.
3. Legacy systems were developed for, and still run mission-critical applications.
4. In their current state, most legacy applications have several challenges associated with their functioning and maintenance.
5. When evolving beyond legacy systems, business leaders must consider strategic issues such as:
. Total Cost of Ownership
. Productivity
. Flexibility
. Knowledge Availability
. Extinct Vendors
. Alignment with Business Goals

6. Various options are available to the CIO when migrating from Legacy systems to more contemporary platforms:
. Functional Extension
. Technical Extension
. Migration
. Replacement

7. Any extension of legacy systems should be "non-invasive."
8. The extension process consists of understanding and documenting the existing system; decomposing the application into data, presentation and processing logic; creating and extracting reusable components; and if desired, converting the legacy code into Web compatible languages.