

Modernization of Legacy Systems: When and How?

Dr. Hans Sassenburg

SE-CURE AG

Lenk, Switzerland

hsassenburg@se-cure.ch

Dr. Lucian Voinea

SolidSource BV

Eindhoven, Netherlands

lucian.voinea@solidsource.nl

Introduction

Should IT executives continue to pour money into their legacy systems, replace, or modernize it? Those responsible for IT systems face this question at the end of each fiscal year. Legacy systems have survived mergers, acquisitions, re-engineering efforts, technical revolutions, industry realignment and so on. These systems, some dating back to the 1960s, remain the core of the IT portfolio, even as companies focus on e-commerce opportunities. Such systems were not developed to address needs that an organization sees today; as a result, they tend to limit the organization's opportunities for growth and market strength. Legacy systems are considered to be potentially problematic for several reasons. Legacy systems often run on obsolete (and usually slow) hardware, and often spare parts for such systems become increasingly difficult to obtain. These systems are often hard to maintain, improve, and expand because there is a general lack of understanding of the system. The designers of the system may have left the organization, leaving no one left understanding or to explain how it works. Such a lack of understanding is increased by inadequate documentation or manuals getting lost over the years. Integration with newer systems may also be difficult because new software may use completely different technologies. Maintaining, replacing and modernizing legacy systems are amongst the most difficult challenges many companies face today. Maintaining legacy systems is often not an option due to the constant technological change. Replacing the existing system by rewriting a new system from scratch will incur several difficulties. Factors working against this approach are for instance:

- Management rarely approves a major investment if the one and only result is lower maintenance costs, instead of additional business value.
- Development of such massive systems normally takes years, so unintended business processes will have to be added to keep pace with the changing business climate.
- Documentation for the old system is in general nonexistent or outdated.
- Like most large projects, the development process will take longer than planned.
- And finally, there's a tendency for large projects to end up costing much more than originally budgeted.

This explains why many organizations choose for modernization as the last option. But what is the impact, what are possible paths to follow?

Legacy code challenges

Legacy systems are often incompatible with interfacing systems or difficult to integrate. A legacy system may be difficult to integrate with other systems like Internet-based business applications. In addition, legacy systems may be monolithic and poorly layered opposed to modern distributed and layered architectures. And last but not least, the code quality of legacy systems in general is poor. Table 1 summarizes some of the most common problems in these three areas.

Problem area	Description
--------------	-------------

Architecture	Common problems include incompatible or difficult to integrate (often proprietary) platforms, fragmented and/or redundant data sources, inflexible architectures which are difficult to change, and insufficient security.
Design	Applications may be poorly layered, with the user interface layer directly accessing the database for example. An asset may be of high quality but its original design goals may not fit current project needs; for example it may be a batch system whereas real-time access is needed.
Code quality	Examples of problems in legacy code are inconsistent naming conventions, inconsistent or missing internal documentation, highly coupled code, low cohesion and high complexity at subsystem and module level, presence of dead code, constant definitions for identical parameters distributed through the code, and code that is difficult to understand.

Table 1. Legacy code challenges.

Also known is the fact, that IT departments find it increasingly difficult to hire developers qualified to work on systems in languages like Cobol no longer found in modern technologies. Driving the need for modernization is the cost versus the business value of legacy systems. A business case analysis should reveal whether increased business value outweigh the investments for modernization.

Balanced Approach

As IT executives select the best modernization path, they must consider the impact on different dimensions related to the legacy system. Together, they represent the system's evolved value to the organization. The additional dimensions besides the legacy system itself are (Ferguson, 2007):

- The legacy system's database (data). A typical legacy system's database has taken years to build and refine. Over the years, business rules and semantic dependencies have been built in at the field and table levels, and these are known to developers and users.
- Business processes. Development staffs have been building and refining systems for years. They know the system inside out, and they understand the business processes that these systems support. The user community is also used to working with these systems, and they too have familiarity with how they work and why they work that way.
- Human capital (staff). Many companies have found, too late, that they had a great deal of investment in their staff (training and intellectual property), which they lost when they replaced systems. Replaced staff takes with them years of knowledge about business, business processes, and supporting systems. This knowledge is locked up in their heads and usually not documented (anymore).

The effects of modernization on human capital or staff are probably most underestimated. By modernizing a system in a way that enhances rather than destroys its human capital, the organization leverages existing human resources and extends their capability and productivity. In the situation where systems are replaced, it takes years to replace the knowledge that has been built into legacy systems and the staff who support it.

Modernization Options

When planning a modernization effort, it should be carefully considered how best to leverage existing assets. And, it must be considered how best to support future initiatives, about which they may yet know very little. Some options for modernization are:

- Screen scraping (or "Web facing") is a method of Web-enablement that has been very popular among users seeking to make applications and data available through a graphical interface or to convert host screens to a Web-like look and feel, without altering the applications or processes. This technique provides Internet access to legacy applications without making any changes to the underlying platform. Because they're non-intrusive, screen scrapers can be deployed in days and sometimes hours. However, scalability can be an issue because most legacy systems cannot handle nearly as many users as modern Internet-based platforms.
- Legacy wrapping is a second option. Like screen scraping, wrapping techniques are applicable in situations where there's no need to change business functionality in the existing platform. The technique builds callable APIs around legacy transactions, providing an integration point with other systems.
- The migration of legacy code to another modern programming environment (like Java or C#) that can be run on modern distributed hardware. This results in true modernization, yielding code that is compliant with modern programming environments and a system capable of meeting both present and future needs. The choice of the new technological environment requires careful consideration. All can create sophisticated, modern business applications that can access a wide variety of database platforms, but they have various effects on the four dimensions.

It is important to consider the ways in which they each leverage and impact each of the four dimensions: legacy, data, processes and staff. And, in addition, it is an important requirement that a modernization path supports future development and growth. Any modernization effort must preferably result in a system that both enhances the organization's agility, but at the same time preserves the four pillars upon which the business depends. Screen scraping (or "Web facing") is an expensive option that is unlikely to return its investment. It changes the appearance without real modernization. It fails to enhance the organization's human capital or staff, as the programmers simply pass the legacy source code through a tool that creates the output. The resulting Web pages are limited in functionality and cannot be enhanced. If any changes to the system are made, all of the affected screens have to be re-scraped. Scraping has no modernizing effect on the data. The legacy source code and systems remain, only the interface is modern. Wrapping does not provide a way to fundamentally change the structure and improve the code quality of the legacy system. Finally, screen scraping and wrapping do not address the high cost associated with maintaining a legacy system or finding staff willing to work on obsolete technology. So, the migration of legacy code to another modern programming environment on modern hardware will in most cases be the most favourable option.

Migration Steps

In the early stages of the migration process, core business logic must be identified and mapped out to show the interrelationships of the code performing the system's business function. Source code analysis can be performed to produce call maps and process flow diagrams, which contain program-to-program call/link dependencies. The resulting call graphs make it possible to visually identify connected clusters of programs, which are good indicators of related business activity.

Once core business logic is identified and mapped, it can be broken up into standalone components deployable on client/server and Internet-based environments. This process creates collections of programs that together perform a specific business function. In addition, the components have clearly defined APIs and can be accessed through modern, industry-standard protocols. Components can remain on a mainframe or be re-deployed into modern, distributed environments.

As part of the transformation process, the common system utility functions such as error reporting, transaction logging, and date-calculation routines, must be identified. To avoid processing redundancy

and to ensure consistency in system behavior, these components need to be standardized into a system-wide reusable utility library.

Conclusion

Legacy modernization is crucial for organizations spending too much time and money maintaining the business value of their outdated information systems. A second factor driving the need for change is the industry's movement toward new Internet-based platforms, such as .NET and J2EE. Adopting newer computing systems can cut operating costs and make it easier to adapt an information system to market changes or competitive pressure.

A variety of options exist, including measures such as screen scraping and code wrapping. Each of the approaches makes sense under certain circumstances. The former can eliminate legacy applications in which the code quality is too poor to migrate, whereas the latter method provides quick and inexpensive access to legacy functionality. But for those companies looking to preserve and extend the functionality of their older system on a modern platform, legacy migration using supporting tools is probably the most cost-effective option. It is a strategic approach to modernization that better positions the organization for the future.