The Business Case for Legacy Architecture Transformation

by William Ulrich

Legacy architecture transformation describes the process of modifying the form, design, and/or function of one or more legacy applications and/or data structures. A legacy architecture transformation strategy defines a commonly agreed upon philosophy that an enterprise can use to reconcile legacy architecture limitations with high-priority, time-critical business requirements.

Many business executives believe that aging legacy architectures will eventually be replaced with new, strategic systems. It would seem logical, so the thinking goes, to simply buy new application systems just as one would buy a new computer. This line of thinking is not unusual for someone not fully apprised of the state of legacy architectures and how these architectures have become intertwined with an enterprise's business model.

Most executives have held the belief for more than two decades that these systems would be replaced. Attempts were made to replace systems on multiple occasions. Initially, 4GLs were going to lead the way. Then it was the CASE revolution. During the early 1990s, IT believed that client/server systems would replace legacy environments. Y2K brought a cold dose of reality. Rewrites were impractical and improbable.

This problem is compounded today because few people can articulate what these systems do. IT has lost many of the legacy skills that developed these systems. This loss of skill extends into the business units as well. Not only are the people who built these systems gone, the users who asked for them to be built in the first place are also gone. Knowledge reclamation of critical software assets is in itself a good risk management policy.

Frustrated with failed rewrite efforts, many executives turned to the *buy* as opposed to *build* mentality. Third-party ERP systems

were brought in during the mid- to late 1990s. Some implementations succeeded in part, while others failed entirely. The problem was that the old systems could not be easily displaced, while legacy data created a huge conversion challenge for implementation teams. ERP systems have now joined the ranks of legacy architectures. Many are written using propriety languages, do not support diverse e-business requirements, and have typically ended up running alongside inhouse legacy systems.

Over the past two decades, organizations have spent billions of dollars in failed legacy system replacement efforts. Global Y2K preparation alone cost a half a trillion dollars, and all that did was fix a date problem. Imagine, for a minute, that an organization had unlimited funding for a legacy replacement effort and no delivery deadline ï¿_ ludicrous concepts, by the way, in today's modern business climate. Such a replacement effort would still fail using traditional rewrite strategies, because the business rules and underlying data could not be replicated through traditional user/analyst re-specification techniques. Organizations no longer know what these systems do.

In spite of all the efforts to replace legacy systems over the years, these systems continue to function and have actually grown in size on an annual basis. Occasionally, a new system is developed to surround or interface with an existing system, but legacy systems are deactivated in only a small percentage of cases.

While politics tend to drive many strategic systems initiatives, including the deployment of a multi-million-dollar ERP application, executives, managers, and analysts should take an objective view of more pragmatic options. Exploring some of the motivations behind failed replacement projects is an important step in this process. Below are some of the rationalizations that have driven high-profile, legacy replacement project failures.

"The old system does not do what we need, so we must build a new system from scratch, which means that IT should not even look at the old system for ideas on how to build the new system."
"User requirements have changed, so we are going to just ask the users what the new system should do. They can re-specify it from

scratch." "A large, multinational consulting firm assures me that a total replacement is the only way to go. If things go wrong, they can take the heat for it." "The old system uses old technology and we need to build a new system using new technology." "The mainframe has to go along with the systems that run on it. I want the mainframe gone by the end of next year." "We need to buy a package and get these old systems out of here." "The best approach is to just leave the old systems alone. We can wrap them with new technology to make them do what they need to do." "New technology, such as Java, XML, and HTML, will allow us to rewrite everything in less time."

Politics aside, allowing the advent or promise of new technologies to drive a legacy replacement decision is not wise from a business perspective. Junior programmers are no more likely to succeed in a rewrite than their predecessors did in prior decades using other "new" technologies.

Conventional wisdom has dictated several evolving schools of thought during the past decade. During the 1990s, the from-scratch rewrite approach was displaced by the ERP package implementation approach. The package replacement approach has since been displaced by the EAI approach. Legacy system wrappers, for many of the reasons stated previously, provide only a stopgap measure. This brings executives back to the table to discuss alternative approaches.

If your enterprise's executive team can bring itself to the point where frank dialog replaces political posturing or the promise of technological miracles, here are some questions that you can ask as a first step in crafting a legacy architecture transformation strategy. Can legacy systems be replaced any time soon by rebuilding them? If this were possible, we would have done that already. Remember that these are the same legacy data and application architectures that businesses and governments spent up to half a trillion dollars just to make Y2K-compliant. They did not replace them then and will not replace the bulk of these systems any time soon - at least not using conventional, re-specification methods.

Will buying and implementing an ERP package help address the

situation? This should be examined carefully because there are many hidden traps in this approach. An ERP system may not conform to your business requirements or may clash with legacy systems and data to the point where competitive advantage is lost - not gained. ERP implementations can also drag on for years, may succeed only in certain business units, and can cost up to 10 times the cost of the software itself.

Can IT use wrapping or middleware tools to access or trigger the data and transactions embedded within these systems in such a way as to make them more agile and adaptable to business requirements? This can be done, but in limited ways. Many Web-based front-end systems already trigger back-end mainframe transactions, but this approach is limited at best. There are many examples where online banking systems cannot update a bank balance in real time, where an order system could not effectively trigger a procurement process, or where other Web-based interfaces failed to effectively complete many of the capabilities they initially promised. Legacy architecture limitations prevent quick fixes and easy answers. A phased approach, however, could incorporate the best notions from each of the above options and be heavily augmented with a variety of legacy architecture analysis, reuse, and transformation techniques. Most legacy transformation solutions are hybrid strategies that incorporate buy, build, and reuse options over a period of time that provides interim value through phased delivery cycles.

The business case for incorporating legacy architecture transformation strategies is based on the process of elimination. If legacy architectures are preventing you from expanding into new markets, delivering new products and services, fulfilling customer requirements, streamlining supply and distribution chains, going global, or just staying competitive, legacy architecture transformation can help. ERP packages, new development, and EAI may all be a part of this strategy, but legacy transformation may be the missing component to make these other strategies work.