Our Struggle to Modernize Applications: How Reuse Wins the War (Again)

The battle to rejuvenate our legacy IT systems is upon us and basic economic lessons can help us prepare

White Paper
# Table of Contents

**ECONOMICS OF REUSE**

- Software Development .................................................................................................................. 4
- COTS ................................................................................................................................................ 4
- Uber-Systems ................................................................................................................................... 4
- Control via Architecture ................................................................................................................. 4
- Shared Services/Consolidated Purchasing ....................................................................................... 5
- Internet/World Wide Web ................................................................................................................ 5
- “Cloud” ............................................................................................................................................. 6

**APPLICATION MODERNIZATION** ................................................................................................. 6

- To-be Environment .......................................................................................................................... 7
The day of reckoning is here. Countless applications have been built since 1951 when the U.S. Census Bureau purchased the first commercial computer, the UNIVAC, from what would later become Unisys. Billions of lines of legacy code are valued in the trillions of dollars. And the plethora of technologies and the complexity of interrelated systems are fast exceeding our ability to manage these mission-critical environments.

Every day our dependency on IT increases and with it our unease, as everything from hackers to high costs threaten the digital fabric of our business and government. And the latest trends – IT consumerization and the “cloud” – will likely exacerbate the business dilemma of aging systems and expanding complexity. As critical IT systems grow more fragile there are no easy answers. Yet CIOs must communicate a credible roadmap – the “to be” – if there is any hope of building consensus for funding needed to avert catastrophic meltdown.

Basic economics – the supply/demand directing our capitalistic system - invariably drives IT. And the short economic history of the information age has everything to tell us about our tomorrow...

Economics of Reuse

Reuse is a universally good thing from an economic standpoint because it almost always makes something more efficient. It usually costs less to reuse something than start over from scratch. Whether talking about recycling, production processes or government information, anytime we can reuse, rather than recreate or buy again, we save. Okay, this is obvious. But this simple fact can provide new and critical insight into the daunting challenge of sustaining our digital fabric. The short history of our IT-based economy, starting in the 1950s, has been an ongoing race for reuse. Time after time, as discussed below, the better reuse strategy was eventually recognized and adopted. The profit-maximizing goals of the most successful companies continue to strive for reuse in order to stay competitive. Understanding this race for reuse can help us prepare for the inevitable transformations as our aging application infrastructure merges with more agile technologies.

Acquisition Allusion: Federal Reuse

If reuse increases efficiency, why is it not explicitly evaluated in procurements, as we do with past performance, for instance? Yes, all procurements include rules about “rights in data” and other intellectual property (IP) rules govern the rights to IP reuse. The OMB-300 capital planning process requires documentation as to whether reuse is included as part of a major initiative. But how much reuse is really occurring in the Federal government? And if a government agency can buy something once and benefit the rest of the government through reuse, how should that value be measured? Simply evaluating reuse strategies, from both a technical and price perspective, in all procurements would dramatically change the vendor culture and lead to more reuse…and savings.
Software Development

Reuse economics has long been recognized as the key paradigm in making application development efficient. Taking advantage of proven, tested, software code lets you move more quickly, more predictably and less expensively each time you reuse. Building in reuse from the start costs more, but can have dramatic benefits. Over the years our most talented architects have come up with innovative ways to promote reuse, such as modular design, object orientation, model-driven development and other techniques. Analysts have variously calculated cost/benefit but the results are the same – reuse makes developers more efficient.

COTS

The earliest days of software development saw a lot of sharing of software code, but as its value was recognized, proprietary business interests came to dominate. Tremendous wealth was accumulated as the dominant reuse strategy arose: Commercial-off-the-shelf software or COTS. COTS revolutionized the IT business model because rather than write code we could buy software that was production ready. Governments enforced policies to make COTS the default over custom code. Reusing software via COTS became the most profitable business the modern economy had ever seen.

More recently, a new form of COTS came to challenge the proprietary model: open source. Open source, at first derided as “shareware,” is becoming a more significant model because it enhances reuse by nature of its more flexible licensing scheme. We need to look no further than the dominant statistics for Linux and the Apache Web Server on the Internet to see how this reuse strategy has competed successfully with the biggest proprietary vendors. And the startling speed with which sales of the open source Android mobile phone operating system has overtaken Apple’s iPhone sales share is the most recent evidence that open source is a reuse model to be reckoned with. In fact, the efficacy of this model encouraged governments, including the U.S. Office of Management and Budget (OMB - M-04-16, Software Acquisition, July 1, 2004), to issue policies to clarify that open source is also COTS, and thus preferred to custom.

Uber-Systems

A variation on COTS that has had great impact on reuse is the mega system perhaps best represented by enterprise resource planning (ERP) environments. While these large and complex systems are COTS, the challenge in “configuring” the systems can be immense. They enforce reuse by defining a full architecture for a domain area. When organizations have been able to adapt to these uber-systems the reuse is very high. However, the risks in implementing large systems is high and the lock-in that usually occurs has the potential to minimize the benefit of the reuse for the customer. COTS by nature will maximize profits and deliver the financial benefits of reuse to the vendor.

Control via Architecture

Another example of reuse policy is the U.S. Department of Defense (DoD) initiative in the late 1980s to make Ada “the single, common, computer programming language” (DoD Directive 3405.1 Computer Programming Language Policy, April 2, 1987). Logically, with all code developed in one language, it would be easier to share software and connect different systems. While it may now seem naive to have tried to control the market in this way, the dramatic DoD reaction helps us to understand the seriousness of the technology problem we still face.

Standardization of the environment has been a goal of many organizations, private and public. To varying extents and depending on the complexity of an organization minimizing the number of technologies has been an indirect contributor to reuse. Mandating the use of one tool, operating system or architecture, ranging from common desktop configurations to middleware, has been tried by many with mixed results. Today, an architecture-driven policy of reuse has taken a more realistic tack towards promoting sustainable system evolution. We have reference architectures, process models, master data management, segment architectures and an ever-expanding choice of
standards. **Standards have likely enabled the greatest amount of reuse.** Think of not having SQL to interact with common databases? Think what would happen to email without IMAP/SMTP? Clearly, without standards of all kinds IT would be nonfunctional.

Yet, while great strides have been made in enabling interoperability, building common process models, and data sharing, we are still faced with growing complexity and limited reuse.

**Shared Services/Consolidated Purchasing**

The most obvious way to get reuse is simply to **have more people use the same thing.** This approach has taken many flavors. In one context we refer to acquisition efforts to consolidate common services by limiting the number of delivery choices. The U.S. OMB promoted this style of reuse beginning in 2004 with significant impact. Solution delivery roles were assigned to lead agencies to consolidate services like financial management or human resource systems, allowing other agencies to simply procure the capability they needed from a competitively awarded short-list. There are obvious **economies of scale** that benefit from this kind of service delivery reuse.

Similarly, consolidated purchasing benefits from economies of scale and encourages reuse. In the U.S., via the General Services Administration (GSA), the SmartBUY program has encouraged reuse by identifying solutions that are widely needed and making them easier and less expensive to buy while encouraging standardization. The challenge is to find flexible market-based incentives to encourage shared services (see sidebar “Acquisition Allusion” for one suggestion) as the procurement burden of picking winners/losers for shared services is high.

Governmental organizations with similar business processes, including state and local governments, still have much to gain from this style of reuse.

**Internet/World Wide Web**

The **Internet**, building on standards, has delivered **massive reuse.** In a relatively short period, the Web has become the ubiquitous interface for computer interaction. The vision of a single, standard, UI environment accessible to everyone seduced developers and policy makers alike. The reality of different versions, proprietary extensions and technologies has dampened this Web browser myopia and given way to a richer environment powered by Internet connectivity. Browser plug-ins like Adobe Flash and non-browser “fat clients” are supplementing the traditional pure HTML thin client. But by making everything more accessible, the Internet makes **reuse just a click away.** Wouldn’t it be wonderful if all of our applications could be magically re-architected to take advantage of Web-style interactions?
Cloud

The “cloud” is really a buzzword for a dizzying array of new age reuse strategies, arguably the most effective strategy to date, and made possible by the Internet. Providing vertical applications, software platforms and/or infrastructure, the cloud promotes reuse in everything it does. Virtualized hardware via infrastructure as a service (IaaS) reduces excess capacity. The most efficient shared applications via software as a service (SaaS) have multi-tenant designs so that all users have the exact same capabilities. SaaS is at its best when delivering standard functionality like email and Web search. The cloud segment defined as platform as a service (PaaS) tends to provide commodity software platforms but is evolving to deliver more value through customization. This emerging area is called APaaS or application platform as a service.

It is interesting to note that many of the powerhouses in this consumer driven market like Google, Amazon and Salesforce, have further leveraged reuse by taking advantage of open-source COTS and open interfaces. The utilization of open source on the Internet builds an even more powerful reuse case, making these firms more competitive. And by enabling an open application interface these firms unleash a torrent of mini-apps or mash-ups that further leverage reuse of the core asset be it geographical mapping or product sales.

Application Modernization

As outlined above the history and definition of success in application development has been directly connected to the level of reuse. Now we face a truly historic software challenge as a large proportion of the code running our mission-critical systems is aging, in some cases directly threatening business success. What can the history of reuse tell us that will help to optimize what will inevitably be a huge investment over time?

App modernization poses some basic questions:

- Can we afford to replace legacy systems?
- Do we need to replace them all?
- How do we prioritize between them?
- How can we reduce redundancy?

Paradigm Shift: Command/Control vs. Market/Consumer

What can we do with this growing IT complexity, composed of old and new? History tells us that the chance of picking the perfect language, architecture, or big-bang system does not work. Nor can we presume our master-plan documents for enterprise data or reference architecture will give us the right path. Consumer-style access and expectations are the new norm for the digital workspace. We want access and information to our work life just as simply as we get it personally. We cannot predict where consumer-driven technology innovation will lead us. Yet we, particularly in large organizations, retain a penchant for believing our current processes can control the digital backbone as simply as we procured software upgrades in the past. But the IT interactions in and between large organizations have become far too complex for a product approach. First, we need to let go of the wishful thinking that we can control these fundamental changes in the IT eco-system. By asserting less control and promoting more collaboration, we can find ways to reuse what is happening in the market. Recognizing and measuring the power of reuse will help light the way.

- How can we future-proof compatibility?

Answering these questions is at the heart of methodologies referred to as application portfolio management (APM). APM uses various metrics to prioritize which applications are most important and ripe for modernization or even elimination. But many organizations’ IT infrastructure has become so large that this top-down approach, while striving to match business requirements, is challenged by the sheer number of applications and the technical complexity. In one DoD organization a multimillion dollar environment with thousands of servers has been purpose built just to
perform integration testing. The point-to-point linkages between hundreds of applications has grown so numerous that testing must be done on the entire environment when one application changes. The APM view of the environment is valuable for investment decisions but the methodology cannot direct day-to-day modernization efforts.

Assuming we can decide what to modernize, how do we do it? There are many styles/choices from refactoring, code conversion, rip & replace, replace with cloud service, and service-oriented architecture (SOA) enablement to name but a few. And, to make the decisions more challenging still, we have ongoing technology upgrades and pressure from the Web 2.0/cloud/consumerization point of view that demand an agility that legacy systems just don’t have.

Above all, application modernization should be seen as an opportunity to achieve greater reuse. The tools and techniques that deliver the greatest reuse will yield the best return on investment (ROI) or total cost of ownership (TCO) over time and should be considered as an overall strategy. This is not a technology choice per se but rather a different style of governance leading to incremental reuse. Focus on reuse strategy rather than an application-by-application approach or battling for the “one” architecture or the best technology.

When deciding to modernize, the key attribute should be whether the updated/new functionality can be reused or whether external reuse candidates exist. This simple question has massive ramifications when applied across large enterprises or the government as a whole. How the reuse potential is bounded – an office or an enterprise or all state governments – is a major determinant on overall efficiency from an economic perspective.

Going back to the future it is easier to reuse smaller blocks of functionality rather than large systems – once called modularity. This approach makes it easier to remove redundant functionality and expose standards-based services. The enterprise SOA model is still the only model we have to incrementally expose shared functionality that arises from multiple domains. The SOA hype of years gone by was fueled on technology sales

and top-down architectural plans that most organizations simply could not implement. A new, governance focused, incremental SOA approach is needed to get the most from reuse when working across large enterprises. And, in no way, can one create an enterprise modernization strategy without a clear vision of where we want to end up.

**To-be Environment**

If we believe in market efficiency, then reuse will win. Reuse has consistently been the deciding factor in all the major trends we see in application development. What is the next step in reuse for applications? The correct answer to this question will then predict the future and help us find a path that will effectively align our modernization strategy.

The pace of change and complexity is leading us not to a place or solution but rather on a journey of continuous change. Continuous change, of course, is what our legacy systems were built to avoid so to succeed we must first enable change and become more agile.

Reuse history tells us that we favor smaller blocks of functionality to overcome complexity and lock-in. By breaking software into smaller functions it becomes easier to eliminate redundancy and enable reuse.

Services that are small enough to deliver a function independently of other services – fine-grained services – delivered via the cloud would optimize reuse as we know it. These services on the Internet can be identified, monitored and secured. Security must likewise be delivered as a service for the ‘cloud services’ model to work. The road forward will focus more on establishing application platforms that enable the journey by reusing governance tools, monitoring services (e.g., quality of service), and security controls.

Adopting services on a broader scale, across the cloud, and in an incremental way building on supportive application platforms has the greatest chance for broad reuse. And if those services are open source, so much the better for reuse.

This is where we end up. More fine-grained services on the Internet (e.g., ‘cloud services’ if we must). This is inevitable because reuse will win, again.