TRENDS

January 10, 2006

Got Legacy? Four Fates Await Your Applications

by Phil Murphy

FORRESTER® Helping Business Thrive On Technology Change

Includes Business Technographics® data

January 10, 2006

Got Legacy? Four Fates Await Your Applications

Several Trends Converge To Help You Rationalize Intelligently, Prepare For SOA

This is the first document in the "Four Fates For Your Applications" series.

by Phil Murphy
with Mike Gilpin and Lindsey Hogan

EXECUTIVE SUMMARY

Organizations have wasted untold millions on purported one-size-fits-all solutions to their legacy application issues: dump the mainframe; rip and replace; move it all to Unix; and, most recently, outsource it all. As they adopted these solutions, IT managers slashed application maintenance budgets to dangerous levels to fund these efforts. Despite those efforts, today, most organizations are no closer to a permanent solution than when COBOL was the predominant programming language and indexed file access methods were considered revolutionary technology. Several recent trends indicate that IT organizations are ready for a break with the past. Service-oriented architecture (SOA) is a viable "future state" target, and IT managers are now admitting that they will keep some legacy applications much longer. In combination, these two factors are driving vendors to offer complementary tools and services to enable application reuse, and they are heightening interest in application portfolio management (APM) tools to create application metrics and visibility into IT activity. The stars have finally aligned to enable knowledge-based application rationalization and modernization.

TABLE OF CONTENTS

NOTES & RESOURCES

Forrester interviewed legacy renewal vendors, service companies, and user companies about application maintenance issues.

Related Research Documents

"Java, COBOL, And Perl Share A Common Problem"
November 11, 2005, Quick Take

"APM Tools Will Reach $500 Million To $700 Million By 2008"
July 22, 2005, Market Overview

"Web-Service-To-Host Modernizes Legacy Application Portfolios With SOAP, WSDL And New Migration Options"
March 21, 2003, Planning Assumption

HOW APPLICATIONS BECAME LEGACY

Think you don't have legacy systems? If you have applications that are more than three years old, then you have legacy applications. Applications written three or more years ago using HTML, XML, Perl, C#, and Java qualify as legacy applications because they share so many attributes with their older siblings — the original authors are gone, they were poorly documented, and as a result, they are now poorly understood, so nobody wants to work on them.[1] Virtually every organization has legacy applications of some form, but the volume of legacy and its corresponding effect on the organization governs what firms do about it.

Streamlining the existing application portfolio is a necessary step for organizations that have been in existence for a decade or more, especially if those organizations intend to evolve to SOA — why carry all that baggage forward to the next computing paradigm? Only some legacy applications will be reusable, and some aren't fit for anything but retirement — but how can you tell which is which? Why waste money maintaining applications that aren't worth keeping? Why not redirect that money to where it will benefit the organization?

Companies don't have enough money for new projects because the existing applications cost too much to operate and maintain. The situation is untenable. How did things get so bad? The truth is that it has been happening for decades and will continue to happen until IT takes corrective action.

Mainframes Drove Business Automation

In the 1970s, large IT organizations built monolithic applications to automate their core business transactions using the best technology available at the time. In the 1980s, IT organizations improved the monolithic applications with online transaction processing (OLTP) and relational databases, and they introduced desktop PCs. In the early 1990s, managers curtailed mainframe development in favor of client-server development, which featured Unix servers and Windows clients as a cheaper, less complex alternative to the mainframe.

Some organizations took things a step further — if Unix was cheaper, then why not migrate everything there? A one-size-fits-all mentality took hold, and organizations thrashed back and forth between a number of solutions, including:

- **Replace it all with client-server, PowerBuilder, or Visual Basic (VB).** Development standards teams announced that all future development work would have to be done in a single language — for example, PowerBuilder or VB — irrespective of the task assigned and how well or poorly suited the new technology was to the task.

- **Dump the mainframe platform.** Unix vendors did all they could to encourage migration away from the mainframe platform — if a little Unix was good, then a lot of it was certainly better.

- **Rip and replace.** Software vendors convinced IT managers to avoid technical obsolescence by ripping out large, installed bases of mature software and replacing them with newer products. For example, they replaced large groups of internally written financial, manufacturing, and human resources applications with packaged application solutions from vendors like Cullinet Software, McCormack & Dodge, and other vendors that no longer exist — in part, a testament to the failure of rip and replace.

Technical Changes Drove Functional Obsolescence

The one-size-fits-all approaches introduced a lot of technical change to IT at considerable expense but provided little incremental business value. Two large efforts followed in the late 1990s: year 2000 remediation and eBusiness.

- **Most firms underestimated the sheer scope of the 2000 effort.** The change itself was easy: Simply expand a date field from six to eight digits. But the coding part is the simple part of any change; the trouble comes in testing. The task of testing every application in the portfolio was gargantuan enough — creating test environments that could simulate a date in the future triggered a number of costly, unanticipated testing setup problems. Suffice to say that actual 2000 testing costs exceeded the initial estimates by several orders of magnitude. Most damning of all was the measurement of success — if the efforts succeeded, then business users would not notice a thing.

- **eBusiness forced more legacy application changes.** The first generation of Web sites were largely used to push marketing and public-relations-oriented information outward to the public. The second-generation Web sites were transactional — they hosted storefronts or other business transactions. Many of the first-generation Web sites were built in total isolation from IT because IT simply lacked the technical and graphical skills. To launch the second generation of their sites, the companies were forced to make significant investments in integration technology to enable the storefronts to trigger legacy transactions running on the back-end systems. Storefront applications needed to invoke legacy accounting, inventory, and order-processing transactions. While these modifications brought business value of a sort, in the opinion of the business, they were to fix an oversight by the techies. The real business value, in their eyes, was in the storefront applications built by others.

The net result is that legacy systems grew well out of sync with the functional needs of the users, and the IT staff lost much of the functional application knowledge. It would be naive to think that these are the only faults of legacy technology, but these issues, more than any others, have contributed to the demise in desirability of legacy systems. It is equally naive to believe that all legacy applications are good or bad: There is a healthy mix of both in most portfolios, but companies have no reliable way to discern the good from the bad and thus no reliable way to evaluate the appropriate fate of legacy applications today. Vendors, trade press, and IT staff all benefit when they tackle new technology issues, dampening much of the remaining enthusiasm for legacy technology.

THE GOOD, THE BAD, AND THE UGLY IN IT TODAY

IT is at something of a crossroads today. With a decade of tumultuous change behind it and application portfolios from several genres as today's assets, few IT organizations know what they own, much less the value of what they own. To remedy the situation, IT organizations must take stock of what they own, what the business needs, and what tools are available to help achieve those goals. In the process of taking stock of today's situation, IT managers will find good news and bad news.

The Bad News: IT Must Ride The Vehicle of Change Or Be Crushed Under Its Wheels

Almost universally, businesses are not satisfied with the levels of service they receive from their IT organizations today.

- **Existing systems cost too much, leaving too little money for new business projects.** The total cost of keeping existing applications enhanced, up to date, and running consumes a disproportionate share of the budget. Forrester's Business Technographics® research shows that North American and European enterprises spend 75% of their software budgets on ongoing operations and maintenance, leaving just 25% for new investments.[2] That is not enough funding for projects that can increase revenue and leverage new business opportunities. The lack of application knowledge is a primary factor in the excessive maintenance costs.

- **IT managers can't align expenditures with the business.** IT managers have no intelligence about IT activity and spending, so they appear to other managers to be inept. The truth is that IT has spent the last decade automating business processes, while ignoring automation of its own activities. If IT is using outdated, manual, inefficient methods, then how effective can it be in spite of itself?

- **Outsourcing is on the rise.** Companies are looking at outsourcing to reduce costs, as underscored by the sharp increase in the planned expenditures for external consulting services. The percentage of companies planning to outsource custom applications work jumped from 34% in 2004 to 44% in 2005.[3]

The Good News: Several Trends Are In IT's Favor

Several trends are paving the way for IT organizations to revamp the way they maintain applications and thereby streamline their existing inventories of applications.

- **IT managers admit they will keep legacy applications longer.** Managers are finally admitting that they will keep applications much longer than previously imagined. What is good about that? The decision is made, and managers know that the flip-flops are over. There is no one-size-fits-all solution, and moving everything to a unified platform flies in the face of the underlying principles of SOA.[4] It is time to take stock, evaluate what is worth keeping, and craft the missing pieces.

- **SOA provides a viable target.** SOA and service-oriented programming are the new Next Big Things with a difference. For the first time, the destination takes a pragmatic approach to the concept of *reuse*, and the journey is a long, slow evolution, not a rapid, big-bang change. SOA promotes reuse at the right level of granularity, in a context that makes sense to IT and the business. For the first time, the desire for reuse comes with an architecture and programming paradigm that supports reuse. SOA is widely believed to be the future state for IT.[5]

- **Vendors are providing third-party tools to expose legacy transactions as services.** Vendors are now providing tools that ease the burden of creating services from legacy transactions, allowing them to be reused in their present state.[6] This permits IT to meet the business needs right away, but it doesn't wholly ignore the architects' concerns. With business needs met, the architects can alter the back-end artifacts to suit architectural needs as a secondary priority, insulated by the level of abstraction provided by Web Services Description Language (WSDL) and simple object access protocol (SOAP). Forrester has dubbed this approach "migrate while you operate."[7] In short, the back end can change without affecting the front end because of the insulating effect of SOA.

- **APM tools are enabling managers to evaluate existing applications.** The auto-discovery capabilities within APM tools construct an inventory of all application artifacts and a knowledge repository that replaces much of the lost application knowledge. The repository acts as a metrics collection and evaluation vehicle that provides visibility into IT activity and allows managers to take corrective action. The analytical views allow managers to take a fact-based approach to evaluating which applications should be kept, modernized, outsourced, etc., even as they afford programmers a significant knowledge and productivity boost.

The Ugly Part: Change Is Difficult

It has taken IT a decade or more to get into this situation, and it unsuccessfully tried the easy way out several times. The only way out of this mess is permanent and fundamental change. IT must turn automation inward on itself, gather some of the data and turn it into information, and use that information to:

- **Manage itself more efficiently.** IT managers are blind to much of the activity in IT, so how can they manage resource consumption that they can't see? They can't align costs to the business, analyze performance data, or determine why its costs are so high.

- **Admit its shortcomings to the business.** Funding for self-improvement won't come easily, but business managers know that IT managers need information to manage effectively. The adage "You cannot manage what you do not measure" is taught to every first-year management student.

- **Craft a plan to move forward that addresses business and IT issues.** With the desire for improvement and the understanding of what is needed, it will be simple enough to craft a plan to organize IT to run more efficiently and communicate activity and expenses more effectively.

During a recent APM conference that Forrester moderated, a senior technical manager shared his approach to garnering support for his application rationalization efforts. Using only rudimentary graphics tools, he drew a system diagram of the applications under his control, laminated the diagram, and distributed it to the business unit managers. On a single sheet of paper, it was a highly complex, ugly diagram. He asked his business unit managers — the owners of the systems — two questions:

- **How do you expect me to understand this mess?** The diagram was quite messy, hard to follow, and conveyed the complexity of the systems, even at the simplest level of abstraction.

- **How do you expect me to tell you what it will cost to change it?** If it is too complex to understand, he argued, then how can estimates bear any relationship to accuracy?

In support of the saying "A picture is worth a thousand words," the diagrams succeeded where countless verbal pleas had failed — the business unit managers empathized with his plight and funded his proposal for better IT metrics to enable better management.

INVENTORY, THEN ANALYZE

IT must change the way it works with the business and embrace the newly available tools and techniques. Admitting the mistakes of the past, in the context of the opportunities and threats of the present, will help leaders clear the air, while crafting a path into the future that all affected parties can accept and support. This may be an oversimplification of the effort, but the steps truly are that simple:

- **Start with an inventory.** You can't streamline an inventory that doesn't exist, so an inventory is the first step. Firms need an accurate inventory of all of the artifacts within IT, as well as the relationships between artifacts, so that when an application is retired or rewritten, all of the pieces can be replaced or eliminated. Manually collected inventories simply get out of sync with reality too quickly to be of value.

- **Zero in on the areas that cause the most pain.** The entire IT operation is not in distress: Certain areas of IT run well, while others remain the bane of management. The labor to maintain custom-developed applications is the bulk of unknown/unexplained expenses in many of Forrester's client companies. Packaged applications see little of this type of maintenance because the company does not possess the source code.[8]

- **Collect inventory data using automated discovery.** Some firms believe that a manual inventory suffices, but because there are so many artifacts and relationships that change so often, maintaining an accurate inventory without automation is impossible. Even in the case of tools that discover runtime modules, rather than source code, the volume is too great for manual methods.

- **Make it continuous.** The inventory must be a living, continuously updated inventory, because new development is a continuous activity. Snapshot views will help you fix the problem this year, but next year (or quarter) you'll have to go through it all over again, and you will miss the opportunity to build on the information throughout the year — that is where the real value accrues. The idea is to build a permanent vehicle to enable you to evaluate applications on demand: when a merger is proposed, when outsourcing is evaluated, when a packaged solution is proposed, etc., with always accurate, up-to-date information.

EVALUATE APPLICATIONS

IT managers have been evaluating applications based on insufficient information for years. The newly available information will lead to the creation of new benchmark comparison statistics and the adoption of existing standards such as those based on the work of Halstead and McCabe, among others.[9]

- **Add metrics to increase the value of the inventory information.** Knowing how many applications you have isn't very helpful. A manual count will tell you the number, but you can't do much with the information. The goal should be to develop information about the applications that will enable you to have business-level discussions with the systems owners about the fate of the application. Metrics should let you discuss the application in terms of staff, money, and time, as opposed to lines of code, megabytes of storage, objects, components, and programs.

- **Assemble views of the information.** The application *benchmark* views will show how a given application ranks against all others in the portfolio, as compared by one or more metrics: cost, complexity, size, maintenance activity, etc. Application *trending* views will show progress against a goal over time, from this year to last year, based on one or more metrics or based on service-level agreement (SLA) achievement, for example. [10]

- **Survey stakeholders.** The opinion of stakeholders is an important perspective on the applications. One can argue that if the system owner really loves or hates the system, then the rest of the metrics may not matter. Firms that decide against automated discovery and continuous updates will have little information except static survey information to analyze. In these cases, develop data from any available sources, which may include compliance to technical standards or stakeholder opinions about functional and user interface features.[11] Combine it with other informational views to increase its value.

CHOOSE A FATE

Organizations that collect data via automated discovery will choose several metrics to monitor, and they will compare the selected metrics across the portfolio or across a large group of applications looking for the outliers: applications that stick out from the pack due to one or more of the metrics.

Cost is an obvious metric, but the application's purpose and the rate of change against it are also important factors to consider. Payroll applications see large amounts of change each year to incorporate tax rule changes, for example. Highly complex applications will be harder to change and are therefore likely to show higher costs. Look for anomalies that can't be explained by some business factor. Applications that remain outliers are candidates for further analysis and will likely be the first to change.

While the full taxonomy of application fates is much more complex, at the top level, the decisions are deceptively simple. There are only four fates that IT organizations can apply to an existing application:

1. **Leave it as it is.** Yes, you can leave some alone — not every application needs fundamental changes to remain useful to the business, and few firms can afford to change every application.

2. **Modernize it.** This category ranges from simple, noninvasive changes that Web-enable to full-scale migrations to other platforms and languages.

3. **Replace it.** One or several applications can be placed under the care of an outsourcing company, whether onshore, nearshore, or offshore. Applications may be rewritten or replaced by a package.

4. **Retire it.** This may be the most overlooked option when it comes to allocating dollars. It takes money and effort to prepare an application for an orderly shutdown, since most cannot simply be left unplugged without sudden adverse circumstances. Conversely, companies continue to run some applications simply because they have always run them — potentially creating myriad reports that no one ever reads. This happens in merger and acquisition activity, when people are afraid of upsetting the status quo. Shutting off such a system is one way to find its users, although it may be a risky method to use.

Fate One: Leave It As It Is

Companies generally budget for a certain fixed level of funding for all existing applications, and so it follows that the fewer applications you touch, the larger the pool of funding there is for those badly needed modernization efforts. So the first question to ask about an application is "Why change it?" not "Why not?" Finite financial resources demand that IT focus on change that will yield maximum positive impact. That leads the questions to a comparative perspective to help thin the pack: "How is this change more important than any of the others?" and "Would the business choose to do this if it could only do one change, or five, or 15?" These discussions also take place for new projects and are the essence of project portfolio management (PPM), but new projects typically have much more exhaustive business cases to support them. Rooting out anomalies in the portfolio often drives maintenance activity.

In fact, the whole idea behind APM isn't to develop deep metrics about every application so much as it is about finding applications with anomalous characteristics or anomalous values in certain key metrics. Applications that show no anomalies are like Newton's first law of motion: They will stay anomalous until acted upon by some other force — a slew of change requests or some other change to the technical environment that destabilizes them.[12]

Fate Two: Modernize It

Modernization covers a lot of territory, depending on one's views about what constitutes "modern." Forrester has divided modernization into several categories:

- **Web-to-host.** These tools work at the presentation or user interface (UI) layer and enable developers to expose 3270 screens in a number of ways: *Green-screen in a browser* is self-explanatory; *Interface re-engineering* converts the 3270 interface to a graphical UI and permits navigational changes to the screen flow; and *screen components* wrap screen functions as Enterprise JavaBeans (EJB), XML, and component object model (COM) components. These tools offer a fast way to Web-enable legacy transactions without needing much in the way of new technology knowledge — it can be done by people with only legacy skills.

- **Web-service-to-host.** Web-service-to-host tools function similarly to Web-to-host, except that the end target is a service. The tools wrap the 3270 transactions in WSDL and expose it to calling programs via SOAP. Some of these tools and the newer releases of CICS work with distributed program link (DPL) technology that expose the CICS COMMAREA as part of the service interface, although only about 20% of the CICS code is written in this manner. Both Web-to-host and Web-service-to-host tools are somewhat misleading in that they are dubbed "noninvasive."[13]

- **Componentize.** This invasive option is mostly a holdover from the 1990s, when it was believed that one could invasively extract business rules from COBOL code and wrap them as COM objects. That option failed to win any financial support, and most companies withdrew their offerings. Some products reappeared a short time later, claiming to convert the business rules to Java as a migration option. That option has fared a little better than the earlier versions, but time will tell how much better it will fare.

- **Other integration options and platforms.** There are a wide range of other options, such as enterprise application integration (EAI), JCA, middleware, and a host of other integration technologies, that are currently in a state of consolidation. Options that we formerly referred to as application, process, and data integration options are coalescing into integration platforms that offer multichannel integration that is more in tune with future technology such as SOA.[14] Virtually all are invasive, in the sense that programmers who use them must have access to the original source code and typically change it as part of the integration effort.

- **Migrate from the platform.** Migration is a complex option. In fact, there are three types of migration. One can choose a platform migration only. A good example of this option is the Micro Focus Lift and Shift option, whereby client companies can move COBOL from a mainframe platform to a Windows/Intel (Wintel) platform. The runtime environment provided by Micro Focus emulates the mainframe production elements, such as transaction handling via CICS and database operations via DB2.[15]

- **Migrate from the database management system (DBMS).** Database migrations have been happening for decades — the earliest were during the database wars, when the relational model "won" the war for supremacy with the other models: hierarchical, network, inverted list, etc. Despite winning the war, many DBMS engines based on nonrelational models are still in widespread use: IBM IMS and Software AG Adabas are good examples. Avoid the knee-jerk decision to migrate away from older DBMS technology simply because it is old. Some firms will have valid reasons to migrate — others will find it a hard sell to the business to fund the migration because there is no business benefit. The cost, duration of the effort, and opportunity cost demand a more careful approach before spending money on either decision.[16]

- **Migrate from the language.** Language migration poses the most risk because it normally requires three migrations —language, DBMS, and platform. For a hint of the issues with automated language migration, try running a textual document through one of the free text document language translation services. Take a document written in another language, and translate it to English. Then read it to see whether it makes sense grammatically and syntactically, and check whether the author's intended meaning comes through in the translated document. Often, they make no sense at all. Although a few vendors offer this service for programming languages and tout a small number of reference customers, clients should approach this option warily.

- **Outsource the operation of the application.** This is distinct from the outsourcing of development, which appears under the "replace it" scenario, because in this case, the application already exists. The outsourcer assumes responsibility to operate the application, enhance it, and support it in all other aspects, relieving the internal IT organization of the duty and perhaps freeing it up for new project work.

Fate Three: Replace It

Replacing an application brings most of the traditional approaches and vendors back into play — the consulting services vendors, the packaged application vendors, etc. Organizations can choose to:

- **Rewrite the application.** Firms can rewrite the application using internal staff and traditional methods, or they can outsource development of the application. This is in contrast to outsourcing the operation of the application, which appears as an option earlier in the document.

- **Purchase a packaged application.** Organizations can opt to buy a package in cases where the function is commonly needed, such as financials, ERP, or human resource functions. Organizations may also opt to customize the package as it is installed. Some niche packages support a limited number of vertical industry needs, such as specialized banking applications and policy and claim applications for insurance, for example.

- **Purchase a hosted application.** This is similar to buying a package, except that the vendor also operates the application for you — either as an application service provider where you use one vendor or a more typical outsourcing arrangement where the application vendor is not the package vendor. Hosted applications are often justification for noninvasive modernization for clients who want to modernize the way the application looks but cannot access the runtime environment.

- **Acquire an open source version.** This option is more of an option for system software; however, applications many be available depending on your needs and definition of application. Consider it something of a hybrid between rewriting and purchasing a package. It will come with some pre-implementation effort, and it may or may not come with vendor support.

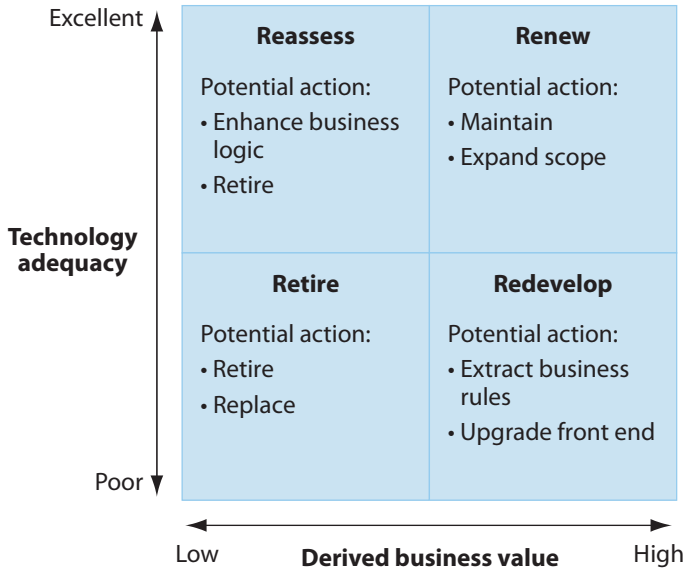
Fate Four: Retire It

The biggest difficulty that people have with retiring an application is that they assume it is a no-work, no-cost option. In truth, there may be considerable effort to unhook an application from all of the places it touches in a mature organization to ensure that those touch points are replaced or plugged and that nothing is broken. More difficult yet is to get people to think about events that the system may trigger once a year, statutory reporting requirements, or data filing requirements where some portions of the application may need to enable inquiry for several years hence. Retirement may not be cheap or easy.

WHAT'S DIFFERENT IN 2006?

What compels a business to pay more attention to maintenance in 2006 and expect different results? For starters, the old approach to dealing with legacy systems is broken.[17] The approach of dismissing legacy applications as things that will some day be replaced has failed — it can't continue because the cost of keeping existing systems running has reached a crisis point. Likewise, the approach that services vendors often use to rationalize legacy applications is insufficient — their clients expect and need more than an evaluation of the applications in light of how large a consulting project each application represents. The depth of analysis tends to be too shallow for today's needs. Their analyses tend to favor two-dimensional graphics to illustrate four quadrants dissected by a horizontal and vertical axis — one representing technical adequacy and the other representing the value to the business or similar metrics (see Figure 1).

Figure 1 Two-Dimensional Views Are Overly Simplistic



Source: Computer Sciences Corporation and Forrester Research

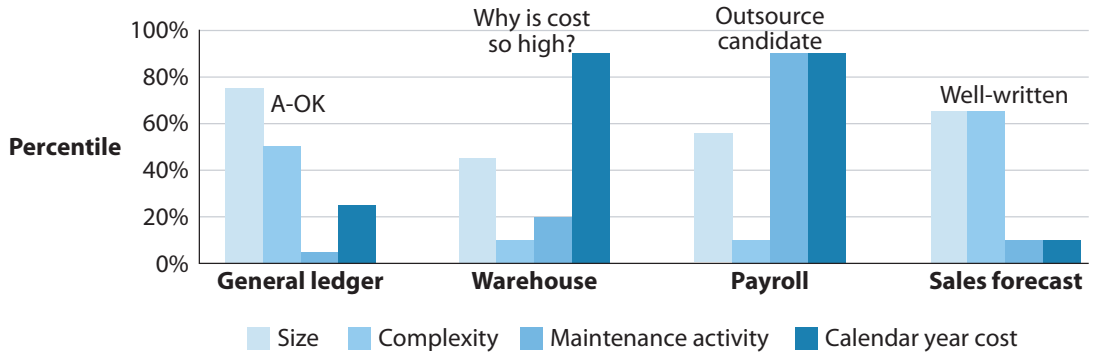
Source: Forrester Research, Inc.

While the mapping of applications to such a graphic can provide some insight, few application decisions are truly as simple as two-dimensional graphics can show. So why use them? In fact, two-dimensional graphics are better suited to illustrating the *results* of a decision-making process than they are to enabling the decision-making process itself. Also, by virtue of the fact that its survey is as of a snapshot in time, the services vendor process does not lend itself to any continuous process improvement.

Firms must do better than simplistic two-dimensional analyses if they are to put an end to spiraling maintenance costs. To reduce costs and gain control of the environment, firms must assemble compound data elements into multidimensional views based on information collected from several sources.

The difference between simple two-dimensional views and compound data views can be astounding. Comparing several metrics at once across a group of applications permits people with little prior application knowledge to arrive at a remarkably well-informed decision about the fate of applications (see Figure 2).

Figure 2 Views Of Compound Data Expose Issues



Source: Forrester Research, Inc.

With just four metrics — application size (in lines of code or function points), complexity, maintenance activity, and cost — it is a relatively simple task to decide the fate of each application:

- **General ledger shows no cause for concern.** Though large and somewhat complex, its very low maintenance activity level and low calendar year cost warrant no further scrutiny.

- **The warehouse application needs scrutiny.** Its average size, very low complexity, and low maintenance activity should translate to low cost. But its costs are well above average. Something is clearly wrong.

- **Payroll is an outsourcing candidate.** The very high rate of maintenance makes sense for a payroll application. Each year, the changing tax laws force coding changes. However, payroll activities are not a core competency of any company — it is commodity business functionality. An application of average size and complexity should translate to low costs. Therefore, hiring a payroll service will likely save the company a lot of money.

- **Sales forecasting validates the move to SOA.** The above-average size and complexity of this application should translate to higher maintenance activity and cost. But this newly implemented application was written using service-oriented programming techniques and architecture — the high amount of reuse is keeping maintenance costs very low, and it validated the decision to move toward SOA.

While the above scenarios are fictional, the value of the compound data views to the application rationalization process is undeniable. Assembling the right metrics into multidimensional graphs enables high-quality decisions about the fate of an application by otherwise uninformed people. Other combinations of metrics will reveal other patterns and issues, limited only by your ability to collect reliable data.

Tool-Side Benefits: Detecting And Mitigating Exposure And Proving Compliance

Some interesting additional benefits accrue as the maintenance process improves. Organizations that have tackled compliance issues, such as Sarbanes-Oxley and Basel II, know that determining whether and where the organization has exposure to the various compliance initiatives is a costly endeavor. It is costly because exposure is buried deep inside the source code of the company's applications. Therefore, organizations need tools that can read and assess source-code-level exposure. Tools that don't read source code to build the application knowledge repository, such as the PPM vendors' offerings, will not be able to assess exposure, determine the size or complexity of an application in any meaningful way, or assess the applications' exposure to various threats (see Figure 3).

IT management needs factual knowledge of what is happening at the source code level. Using the tools, managers can:

- **Highlight exposure to compliance issues and other IT risks.** The tools can discover which applications modify financial data for Basel II and Sarbanes-Oxley and which alter patient-confidential data for HIPAA, for example.
- **Ensure complete mitigation of the exposure.** The tools can identify the artifacts that must change to enable remediation and mitigate exposure.
- **Provide a proof tool to auditors.** The tools can prove due diligence with regard to risk management and compliance initiatives through the ad hoc reporting and real-time views into the portfolio. An organization that can demonstrate its efforts to mitigate exposure in this way offers powerful proof points to any subsequent audit.

The tools are peerless in their ability to help demonstrate compliance and risk mitigation efforts, over and above the value they bring to application rationalization efforts.

Figure 3 The Sources For Application Metrics

Opinion-based metrics: qualitative and subjective to moods and biases of interviewees.

	Source	Surveyee	Comment
Value to organization	Survey	Management	Revenue based, number of users affected during outages, etc.
Functionality	Survey	Primary users	Scale of how well it accomplishes the functions (1-10).
User interface	Survey	Primary users	Visual appeal of presentation and access aspects.
Stability	Survey	Operations	Ratio of number of failures to number of times run or an equivalent ratio.
Match with future technology	Survey	Architects	Scale of alignment with SOA (1-10).
Planned changes	Survey	Management	Backlog of change in business, regulatory, or technical — expressed in person years or another measure.

Fact-based metrics: factual and quantitative but limited to data that can be collected automatically.

	Source	Surveyee	Comment
Maintenance costs	Tool	Calculate	Formula of effort multiplied by average cost of staff.
Size	Tool	Read code	In lines of code or function points, used in conjunction with complexity and other metrics.
Complexity	Tool	Read code	Use industry-standard Halstead, McCabe measures.
Reason for change	Tool	SCM	Use bug-fix, enhancement, new feature, etc.
Number of changes (volume and frequency)	Tool	SCM	Express in volume, frequency, and effort in terms of hours.
Exposure to Sarbanes-Oxley	Tool	Read code	Yes/no based on impact analysis at the artifact level.
Exposure to HIPAA	Tool	Read code	Yes/no based on impact analysis at the artifact level.
Exposure to open source	Tool	Read code	Yes/no based on impact analysis at the artifact level.

Source: Forrester Research, Inc.

RECOMMENDATIONS

GET MOVING ON A NEW APPROACH TO MAINTENANCE

The days of simplistic, one-size-fits-all application decisions are over; they did not suit organizations well in the 1980s and 1990s, and they are an even worse option today. Organizations need a different attitude, approach, and tool set:

- **Forget the rip-and-replace approaches of the past.** History has proven that there are no one-size-fits-all answers. A good approach for one application will be a poor approach for several others. Each application must be evaluated on its own merits and faults. Rip and replace may be the right answer for a few applications, but you won't know until you have better information.
- **Don't rush to judgment; develop better sources of information first.** One of the biggest problems with the old approach is that it based momentous decisions on too little information, and much of the information was subject to the moods and whims of the people surveyed. While stakeholder opinions are important, that type of subjective information needs to be balanced with factual metrics about the application for a complete picture.
- **Invest in automatic discovery — there is too much data to collect manually.** If it were possible to collect data manually, then we would be drowning in custom-developed rich graphics about our applications. But manual collection provides garbage data that no one believes. Look at the tools, fund a purchase, and move on to a decision that warrants debate — this one does not.
- **Get some help.** Change in IT always affects three things: people, process, and technology. Simply buying a tool (technology) and expecting miracles is sure to disappoint. Address the people and process issues, as well. Interview services firms that have firsthand experience implementing APM tools for their clients. Talk to reference customers about how they organized and what issues they faced. Insist that service vendors leave behind a repository and continuous data collection vehicle that can drive next year's application decisions, even as they reduce today's maintenance burden.
- **Organize for success.** The ultimate place to decide cross-organizational issues is a *program office* or an *office of the CIO,* if one exists.[18] That type of organizational structure will have the political power to take on powerful adversaries that lose resources as priorities shift to align with the needs of the business, instead of pet projects.
- **Build a continuous process, not a single event.** Change to applications is frequent, and even in organizations with only moderate levels of change, the metrics also change continuously. Companies add new applications all the time. New applications require integration with the old. Cost and complexity increase with maintenance and integration efforts, underscoring the need for tools.

- **Leverage knowledge for compliance benefits.** While no tool that increases application knowledge should sell itself primarily as a compliance tool for Sarbanes-Oxley and HIPAA, the benefits are undeniable. Without a doubt, the knowledge of the code base will increase confidence in the efficacy of compliance initiatives and security measures, reduce the cost of compliance, reduce exposure to various kinds of code-based threats from malicious activity, and aid proof-of-compliance and due diligence audits.

WHAT IT MEANS

VENDORS AND END USERS WILL BENEFIT

The new millennium brings new tools and new approaches that significantly increase the value of existing applications, even as they reduce the risk of threats to the corporation from compliance-related lawsuits and malicious software attacks. IT organizations must adopt the new approach to maintenance, not only to achieve those benefits but to avoid wasting millions of dollars in vain attempts to force one-size-fits-all solutions on heterogeneous problems.

- **Preparation for SOA will force companies to streamline their portfolios.** In preparation for SOA, firms will increasingly streamline what they own and jettison technology that can't be made compatible with SOA. The cumulative cost of keeping older systems operational in the face of compliance and privacy issues will stifle companies that fail to streamline.
- **APM vendors will be a significant beneficiary of the move to streamline.** The success of the APM tool vendors rests on how strongly the need to streamline takes hold. With no clear leader today, the market is wide open for a capable company with good technology. New entrants to the market, from source code management ranks, from services firms, from PPM vendors, and from vendors seeking to flesh out integrated IT management offerings, will heat up the merger and acquisition activity in the APM space.
- **Services firms will change their offerings to include tools will benefit.** A handful of IT services firms are already shifting their offerings to include more automation in the streamlining process, realizing that to attract the attention of client companies, they need to offer more application choices than rewrite it or outsource it. Large-scale legacy modernization engagements must leave a tool behind to capture the metrics that will drive the subsequent year's activities. This leaves consultants another reason to return — an annual health checkup and a strategy planning session for the new year. This new approach represents a huge incremental benefit to client companies over the traditional "legacy transformation" offerings and requires only a slight change on the part of the IT services firms.[19]

ENDNOTES

[1] Applications like this aren't really legacy, but they do share legacy characteristics — they share knowledge loss, and knowledge loss increases the cost of keeping an application operational by a large margin. See the November 11, 2005, Quick Take "Java, COBOL, And Perl Share A Common Problem."

[2] On average, 75% of firms' IT budgets go to ongoing operations and maintenance, as opposed to new investments. In September and October 2005, Forrester surveyed 911 North American and European software and services decision-makers. Source: Business Technographics® November 2005 North American And European Enterprise Software And Services Survey.

[3] Global-size companies (those with 20,000 or more employees) plan even more outsourced applications work, with 57% expecting to spend on external consulting. In very large companies (5,000 to 19,999 employees) and large companies (1,000 to 4,999 employees), 44% and 38%, respectively, plan external consulting expenditures. See the December 15, 2004, "2005 Enterprise IT Outlook."

[4] A service is a concise unit of work that has finite boundaries and is designed to be very loosely coupled with its environment but is callable from any place that needs the service. WSDL defines the service, and SOAP is the language (protocol) that invokes it. With these characteristics, it can reside virtually anywhere that participates in this paradigm — mainframe, Unix, Linux, or Windows/Intel-based servers.

[5] SOA approaches reuse at the business-transaction level of granularity; "add a customer," "add an order," and "check back-order status" are examples. Since many of the older CICS transactions were written at a similar level of granularity, there is better synergy between SOA and legacy than many people suspect.

[6] Web-service-to-host tools have enabled a number of companies to expose transactions that many believed were trapped inside of the legacy technology as services that can be invoked from a Web front end or any other program needing that service. See the December 6, 2004, Quick Take "Service-Based Transactions From CA-IDMS."

[7] "Migrate while you operate" refers to the ability to avoid the myriad problems created by "big bang" migrations by instead migrating only several services at a time. See the March 21, 2003, Planning Assumption "Web-Service-To-Host Modernizes Legacy Application Portfolios With SOAP, WSDL And New Migration Options."

[8] Unless they possess and change the source code for the packaged application, organizations with a prevalence of packaged applications do not suffer from the high cost of custom-coding maintenance. The costs in these organizations are more likely to come from multiple configurations and deployments or from custom-coded integration points. See the August 8, 2005, Quick Take "Packaged Applications Are Poorly Served By APM"

[9] Halstead's and McCabe's published theories on the topic of software measurement form the basis for much of the available research on the topic. See www.sei.cmu.edu/str/descriptions/halstead_body.html, and see www.sei.cmu.edu/str/descriptions/cyclomatic_body.html

[10] APM is a budding process and technology that inventories IT applications and develops application metrics to permit IT managers to streamline inventories and reduce overall costs. See the October 20, 2005, Best Practices "Building The Business Case For APM."

[11] Firms can evaluate whether to perform certain application activity in the light of technical standards and opinion surveys. Although the results are highly subjective and lack any factual basis, they are better than nothing. See the May 6, 2003, Ideabyte "Evaluating The Fate Of Applications? Think Business, Technology, Core Competency, Opportunity Cost."

[12] Newton's first law of motion is often paraphrased in the following manner: "An object at rest will tend to stay at rest and an object in motion will tend to stay in motion until acted upon by some other force."

[13] Noninvasive is a misleading term — these tools are called noninvasive, meaning that they do not require a change to the original application's source code. Therefore, they can be used in organizations that do not have access to source code or that do not have permission to change it, as in an application service provider (ASP) model. Noninvasive does not mean that there is no coding at all; the Web-service-to-host tools will require front-end application programming to invoke the services, although the interface re-engineering and green-screen-in-a-browser do not.

[14] Literally dozens of integration option technologies in hundreds of combinations may be used to modernize existing applications. See the December 22, 2004, Market Overview "Integration Landscape 2005."

[15] The Micro Focus Lift and Shift offering permits companies to move CICS/COBOL/DB2 applications to a Windows environment and save substantial amounts of money on the platform. See www.microfocus.com/solutions/liftandshift/.

[16] Opportunity cost, in this context, indicates that DBMS migrations require a substantial investment in time, people, and money — what other projects (opportunities) will not be accomplished because managers decided to attempt this migration?

[17] Some services vendors have been selling legacy transformation services to clients that are little more than thinly veiled prospecting for future business. See the June 24, 2004, Trends "The Legacy Transformation Business Is Dead."

[18] Decisions that affect all applications must be drawn up into the management hierarchy so that they are made to benefit the greater good — in the interests of the company, not a few business units to the detriment of others. Commonly, this is a program office or the office of the CIO. See the September 22, 2005, Trends "Taking Your PMO To The Next Step: The Office Of The CIO."

[19] The traditional approach to legacy transformation is a stakeholder survey that, as its primary goal, searches out the largest consulting projects. That may or may not be the client's No. 1 priority. See the June 24, 2004, Trends "The Legacy Transformation Business Is Dead."

FORRESTER®

Helping Business Thrive On Technology Change

Headquarters

Forrester Research, Inc.
400 Technology Square
Cambridge, MA 02139 USA
Tel: +1 617/613-6000
Fax: +1 617/613-5000
Email: forrester@forrester.com
Nasdaq symbol: FORR
www.forrester.com

Research and Sales Offices

Australia
Brazil
Canada
Denmark
France
Germany
Hong Kong
India
Israel
Japan
Korea
The Netherlands
Switzerland
United Kingdom
United States

For a complete list of worldwide locations, visit www.forrester.com/about.

Forrester Research (Nasdaq: FORR) is an independent technology and market research company that provides pragmatic and forward-thinking advice about technology's impact on business and consumers. For 22 years, Forrester has been a thought leader and trusted advisor, helping global clients lead in their markets through its research, consulting, events, and peer-to-peer executive programs. For more information, visit www.forrester.com.

FORRESTER®