3-01-75 Managing Legacy Assets
Christopher Slee
Malcolm Slovin

Payoff
The magnitude of legacy expenditures and their business impact challenge IS managers to both increase the value derived from legacy assets and free up resources for supporting new applications. A coordinated approach that integrates legacy perspectives, processes, applications, and infrastructures with the business-driven development needed in today's hybrid computing environments helps IS managers meet those challenges.

Problems Addressed
To discuss the legacy issue is to raise fundamental questions about how the Information Systems organization optimizes the use of information technology in the business. Instead of viewing legacy applications in purely technological terms, managers must look at the processes that generated these applications in such a way as to make them problematic. This shift in perspective requires that legacy issues be understood within a larger business context.

In reality, IS is simply a microcosm of larger economic forces. Today's businesses are striving to respond to rapid change by creating flexible, modular business processes that can be quickly implemented and reengineered as necessary. To support these types of processes, IS organizations are moving beyond traditional transaction processing into new areas and deploying rapid development and client/server architectures. But if the IS function is merely trying to learn to do systems development with new technologies, it is missing the larger challenge.

Although emerging technologies such as objects, client/server, and the Internet bring exciting opportunities to create new types of business solutions, it is legacy systems that hold the major key to future success. Legacy applications dominate IS resources, represent trillions of dollars of investment, and present the leading obstacle in more than half of reengineering efforts. In many cases, the IS organization itself is a legacy issue, tied to mainframes and old-world processes in a time of rapid change. Faster business change and faster technology change challenge IS managers to integrate business-driven development with legacy perspectives, processes, applications, and infrastructures.

The Legacy Challenge
Recent estimates show that the current installed base of mission-critical legacy applications would cost $3 trillion to replace. Because of the magnitude of these expenditures and their business impact, IS organizations must move carefully yet decisively. Even basic maintenance of legacy systems costs huge sums of money, leaving managers with only a small fraction of their budget for critically important new applications.

These cost pressures will intensify in the future for three main reasons: first, CEO have invested in Information Technology for many years and are demanding accountability for results; second, budget increases have been limited to the rate of inflation for most of the recent past; and third, in an increasingly competitive world, companies continue to demand cost reductions and better returns on investment from every segment of the business. As a result, several challenges confront the IS organization:
At least 80% of most IS budgets is spent on legacy systems, with a declining residue left for new development. At the same time, each new application is more complicated, costing more to develop and maintain.

Because much of the focus of legacy maintenance is on corrections, minor enhancements, and preventing catastrophe, legacy investments quickly reach a point of decreasing marginal utility: each additional dollar buys fewer and fewer benefits, especially when compared to each dollar spent on new solutions to business problems.

The accelerated pace of technological change means that systems are superseded more rapidly than they were when technologies were more stable.

As Exhibit 1 illustrates, too much of the effort expended on legacy systems is fundamentally non-value-added; studies show that more than 50% of the effort is devoted to understanding a legacy application and then retesting the application after a change.

As a result of the intensive effort made to change a legacy application, about 25% of legacy time is spent on testing. A more intelligent, planned, and focused approach should provide adequate testing with less effort.

This year's new application becomes next year's legacy, typically increasing the installed base of legacy systems that must be maintained.

**Legacy Maintenance**

These challenges have several implications for IS managers. They must break the cycle of an ever-increasing yet fractured installed base that drives an ever-larger annual maintenance expense. New systems have to be funded by gains in IT productivity and an accumulation of small cost savings rather than by increasing infusions of corporate capital. As with many business reengineering efforts, IS transformation will have to be quickly self-funding.

**Improving Legacy Management**

These pressures all point to better legacy management as the key to the future of the IS function. IS managers and their staffs should improve their approaches to the legacy issue—the applications, the organization, and the linkage to the business—both to increase the value derived from legacy assets and to free up resources for supporting new applications.

Instead of viewing the so-called legacy problem as a monolith—and despairing that it cannot be defeated, only dented—some IS managers are reframing the legacy issue and helping to drive the strategic integration of business performance and IT. They have begun to address the legacy puzzle with the following series of phased responses:

- Focusing on portfolio management by developing a comprehensive understanding of and strategy for IT assets and investment that relates both legacy and new assets to business processes and the business value derived from each element.
Embracing the challenge of hybrid computing and working to recover value from legacy assets by converting and interfacing the old and new architectures concurrently running in most companies.

Transforming the IS function with legacy considerations in the forefront. This involves rethinking the day-to-day processes and procedures that support the evolution of IT assets; it essentially means reengineering IS itself.

Portfolio Management of IT Assets

Yesterday's sequential planning approach, which provided the IS function with sufficiently long lead times to meet business objectives, no longer exists in today's rapidly changing business world. Although business change drives the integration of process and technology, ineffective communications are causing the gap between business strategy and IS delivery to continue to widen. As IT becomes a major engine of change, it redefines what is strategically possible and gets further embedded into every business process, including electronic commerce, computer-integrated manufacturing, investment program trading, and supply chain optimization. The future of IT therefore requires a holistic understanding of business strategy and processes.

In stark contrast, many decisions regarding legacy systems are episodic, tentative, and trapped in a pattern of inertial spending. To address these tendencies, IS managers should evaluate legacy systems in much the same way as sales territories, personnel, or the corporate image are evaluated. Looking at IT assets as business resources provides IS managers with two main evaluation criteria:

- How effectively does a given system or suite of applications support business objectives?
- How efficiently does the system perform its support tasks?

This orientation brings out a pair of key points. First, both business and technical perspectives are necessary to assess a portfolio. Business objectives are derived from evolving business processes and evolving strategy, and this can be a difficult process. IT assets are mapped against these objectives and evaluated using a comprehensive set of metrics.

Second, the portfolio must be examined in meaningful pieces. Individual elements of the portfolio fit together like puzzle pieces to support a critical business area or process. Even though individual elements may appear to be valuable, the overall business results may be considerably weaker because of critical shortcomings in the way the various parts fit together.

The results of this assessment are then mapped within a grid called the 4R portfolio assessment matrix (see Exhibit 2).

Portfolio Assessment Matrix

The 4R Portfolio Assessment Matrix

The 4R portfolio assessment matrix contains four categories with suggested actions—the four Rs—that guide decisions. Sorting assets into a differentiated portfolio reduces the
Each application is represented by a pie chart indicating its relative size, value, and the recommended allocation of future resources.
magnitude of the legacy problem with a strategy of divide and conquer. IS managers who understand the relative scale, size, and investment in each of the applications acquire insights that guide future action.

**Category 1: Low Business Value, Low Technical Condition; Action: Retire.**

If a system performs a function of questionable value unsatisfactorily, why is it running in the first place? These systems, which constitute about 25% of the North American legacy base, are excellent candidates for early retirement or benign neglect. If they must be kept alive, IS managers should consider installing a graphical user interface Graphical User Interface on top of the character-based screen. Selective system improvement is another option, but only if the cost can be justified with business results.

**Category 2: Low Business Value, High Technical Condition; Action: Reassess.**

IS managers should reassess why a high-performance system is contributing so little to the business. These systems, only about 5% of the installed base, may not have been well justified with an adequate case for action. Alternatively, over time a justification may have become outdated. In other cases, rollout may have been mishandled. IS managers should consider moving these assets to more critical applications if they still are capable of providing business value and a phased retirement if they are not.

**Category 3: High Business Value, Low Technical Condition; Action: Redevelop.**

About half of legacy systems fall into this category. These systems may require pampering, yet the business still depends on them. IS managers should strive to maintain business support, retain asset value, and improve functionality where appropriate—all while reducing cost. This can be done in many ways, for example, by extracting business rules from operational systems, developing value-based cases for action that support replacement, or developing a strategy for gradually substituting new functionality for old.

**Category 4: High Business Value, High Technical Condition; Action: Renew.**

About 20% of legacy systems are in this ideal asset state of delivering tangible business value and being in good technical condition. It is an unfortunate truth that most applications started off in this quadrant or were targeted to start in this quadrant but have since fallen out of it. The organization's mission regarding these systems is to preserve asset value by allowing them to migrate forward as business goals and technologies change.

After assessment, IS managers can look beyond the immediate and see the problems with legacy systems as products of root causes. Rather than viewing legacy applications as fragile and redundant, the IS function can move to extend the useful life of these legacy assets. By addressing the causes of these problems—which are often not technical issues—IS managers can move to prevent the all too-rapid decline in business value and technical condition experienced with most applications.
Institutionalizing Portfolio Management

For all of its benefits, portfolio assessment as a one-time event will not deliver lasting gains. Although it identifies some improvement opportunities that IS managers should act on promptly, over time a one-time assessment rapidly degrades into architectural shelfware. Instead, portfolio assessment must be the first step in a process that lets IS managers continually manage the portfolio by reevaluating the linkage between IT assets and the evolution of business needs. The life span of current technologies is a fraction of that of mainframes; expecting an assessment to drive a three-year plan inevitably leads to mounting questions about its relevance and growing disappointment with its effectiveness.

A well-structured portfolio assessment should lay the groundwork for this ongoing management approach. In addition to the 4R profile, the assessment should address other key questions:

- How do end-user and formal IS-supported solutions interact?
- How do current IT assets support fundamental business goals and processes?
- What is the transition and development approach for each application in the portfolio?
- What is the life cycle condition and investment posture for each application?
- How does the applications strategy fit with the technical infrastructure?
- Where should IS focus its spending?

Answering these questions provides the foundation for a new relationship that aligns the IS function and its business partners around business goals. Portfolio assessment becomes the first step in a fundamental shift away from the IS function's previous pattern of responding to a stream of requests and associated expenses driven by demands for change to existing systems or proposals. Instead, the IS function builds anew toward managing the evolution of a series of IT assets. In this new approach, IS managers evaluate changes against the state of IT assets and the business processes they support.

The assessment itself represents a high-level plan for the development of IT assets; at the very highest level, it is a conceptual architecture for the corporation. A relevant parallel to this high-level plan is found in the model used for citywide planning and construction. Allowing for adaptations resulting from the intangible nature of computing, business processes, portfolio strategy, technical infrastructure, and code inspections all have simple equivalents in the model's basic planning and construction disciplines (e.g., zoning plan, infrastructure plan, design approval, and various permits, code inspections, and maintenance regulations). What the IS function typically lacks is the discipline of the process, the role of a city planning department, and the maintenance regulations for ongoing upkeep.

Adjusting to Hybrid Computing

Some IS organizations narrowly view the legacy challenge primarily in technical terms. In these times of hybrid computing, IS managers must navigate between sharply contrasting world views: relational versus hierarchical, flexible versus rigid, object-based versus procedural, distributed versus centralized, open versus closed. The days of the single
paradigm are gone, and the accelerating pace of business and technological change requires the IS function to accommodate multiple architectures, languages, and platforms. In response, successful companies are rethinking their fundamental approaches to integration, planning systematic value recovery from legacy assets, and devising comprehensive transition strategies.

Rethinking Integration

Legacy Overintegration.

The typical business application has increased in size by an astounding 5,400% since the start of the 1980s. A typical mission-critical integrated application has grown to include 1.2 million lines of code—assembled by stringing together what would have been 20 different applications 15 years ago. These numbers begin to illustrate the challenge posed by legacy overintegration.

Spaghetti code—the dominant challenge 15 to 20 years ago—has been replaced by spaghetti integration. Many architectures present numerous opportunities for uncontrolled interactions among applications, programs, code, and data. Structured programming—a widely accepted step forward over spaghetti code—offers some lessons for spaghetti integration. It stresses tight cohesion (i.e., keeping highly related functions together) and loose coupling (i.e., minimal connections between functions).

Unfortunately, in most cases of spaghetti integration and integrated shared data bases, the opposite is true. Legacy applications are usually characterized by loose cohesion, with functional logic such as product edits and customer data sprinkled through numerous programs and applications. The applications are tightly coupled both through shared data bases, redundant data bases, and interface files and through uncontrolled interactions between numerous fields in countless programs.

Shifting to Component Architectures.

Component architectures replace shared data integration with tight cohesion and controlled coupling—practices that have much in common with object-oriented design and analysis. These architectures shift the focus to standardized interfaces and construction guidelines. In addition, communications mechanisms based on the use of components complement the traditional focus on applications and data. Component architectures use such items as desktop integration, a software message bus, remote data access, and data warehouses as building blocks to help applications cooperate through standardized interfaces.

Although the idea of components is readily understandable in terms of new development, it can also be applied to legacy applications. Some organizations are now viewing these applications as reusable components that can be incorporated into new development using object-oriented techniques. Building on component-based message-driven architectures, IS professionals can use a variety of techniques to avoid plunging into the heart of legacy systems and forcing in new levels of complexity. Many of these techniques are now becoming well established, as the following sections on value recovery and transition strategies will discuss.

The shift to component architectures also supports two concrete commitments—reuse and maintainability—that increase the value of IT assets. Reuse increases the value of an asset by reducing future development costs. Maintainability increases value by allowing an
asset to evolve as the business changes. Both significantly reduce costs by focusing investment on fewer, more flexible assets. Breaking applications into controllable, standardized components is as important to maintainability as it is to reuse.

### Recovering Value from Legacy Assets

Legacy applications are storehouses of immense business value, even if that value can be extremely difficult to exploit. The value can be classified as falling into one of three main areas: data, processing logic, and business rules.

Fortunately, new strategies, techniques, and tools are emerging that let IS managers and their staffs reengineer, recondition, coexist with, or extract value from existing applications. These approaches also move legacy assets closer to the architectures that underlie new development.

### Transition Strategies for Legacy Assets

Several fundamental techniques are now emerging as the basis of a value recovery program in a hybrid environment. As the following sections illustrate, most of these tactics rely on breaking large applications into smaller components and then reengineering or reconditioning them.

#### Reverse Engineering.

Automated tools help raise a system to a higher level of abstraction by, for example, deriving a system specification or requirements model from existing code and data. This technique may create a new baseline for incorporating enhancements and enabling future code regeneration. It also facilitates traditional maintenance by enhancing developers’ understanding of the system.

#### Package (i.e., Components).

Standard Commercial Off-The-Shelf (commercial-off-the-shelf) packages such as word processors, spreadsheets, workflow engines, and graphics libraries can provide critical components for hybrid solutions.

#### Components.

Partitioning legacy systems or legacy programs into smaller components lets IS staff phase out or replace a system one piece at a time. Some code analysis tools help with this task, but the conceptual commitment to modularity and reuse is more important than particular tools.

#### Rationalization and Restructuring.

Cleaning up existing code by eliminating redundancy, instituting standards, improving structure, and updating documentation simplifies maintenance and enhancement. This is frequently the first step in a legacy strategy.
Conversion and Rehosting.

Rehosting involves moving legacy applications—untouched—onto client/server platforms. This approach not only promises potentially lower costs but also enables multiple application components to be more seamlessly integrated into a solution for the business user.

Architectural Layers.

Layering separates the various components of an application, such as presentation/user interface, application logic, data access level, and communications.

Wrappering.

Creating a software wrapper to encapsulate and modularize legacy components enables them to coexist and communicate with object-oriented components. A function server lets the legacy code send and receive object-oriented messages. In this way, wrappering positions legacy code to provide continuing value and to be reused in future systems.

Forward Regeneration.

Automated tools help developers regenerate code based on modified higher-level abstractions rather than modifying the code directly. This technique may be used after reverse engineering, or if the original system was developed through code generation.

Surround.

Creating additional functionality and data components around the legacy system—but without modifying the system itself—allows legacy components to be phased out as the surrounding components supplant the legacy functions.

Data Warehouse.

Whereas surround strategies put new functionality in front of legacy systems and integrate at the desktop level, warehouses integrate the data behind legacy applications.

Maintenance and Enhancement.

As the section on transforming the IS organization will illustrate, leading practitioners are rethinking the traditional approach to managing the evolution of legacy assets even in the areas of maintenance and enhancement.

New Development.

Best practice models and capability maturity assessment help chart a course to better results and productivity in new applications development.
Package Replacement.
Software packages often replace a legacy system, especially if it is used for nondifferentiating basic support applications. COTS application packages may also be used in a surround approach.

Outsourcing: Long-Term or Transitional.
Because, it can be cost-effective under the right circumstances to identify a clearly definable task and outsource it, IS managers should consider outsourcers in the enhancement, redevelopment, or replacement phases of a legacy strategy.
These transition techniques are certainly not standalone mechanisms. Not only is there overlap among the various techniques, but they also interact with each other to produce more finely tuned results. IS managers should carefully assess the techniques in a given situation as part of the overall project plan.

Transforming the IS Organization
Once the IS organization is committed to developing a system of architectural components, IS managers face the task of transforming the way the IS function itself interacts with the legacy portfolio. Because IS managers have relatively stagnant budgets, improvements must be rapidly self-funding. Such improvements result from five sources:

- Better alignment of business and IS goals and strategies.
- Better partnering between the business and IS communities.
- Better management and technical processes within the IS organization.
- Better skills, practices, tools, and techniques at the developer level.
- Continuous enhancement of IS capabilities.

Transformation of the IS organization centers on managing user demand, legacy systems evolution, and IS resources. There is a tendency to accept an IS organization’s behavior as yet another legacy rather than to challenge old assumptions and reshape managerial approaches. The IS function must fundamentally rethink and reengineer itself in the same way that many organizations are reengineering business processes.

IS management has traditionally focused on significant technology changes that cause work processes, skills, and understanding to lag until they are reshaped. Recent examples of such change include client/server technology, object orientation, and multimedia. The fundamental issues of IS transformation, however, relate to commonly accepted practices and beliefs—in other words, to the culture within IS itself.
IS Cultural Change

Craft-Based Development.

Much IS activity is still conducted in the cultural equivalent of a preindustrial craft. Methods are personal and unstandardized, private knowledge confers power, and success or failure is not readily determined. At worst, code is written in idiosyncratic ways by employees who must be retained to update their personal handiwork, trapping other developers because nobody else supposedly understands that system. Finally, a member of the guild can only be understood and therefore judged by another craftsperson, who is hesitant to criticize comrades. For example, even though formal peer review is proven as the most cost-effective method for preventing and eliminating coding errors, few IS developers inspect each other's work.

A growing body of evidence highlights the severity of this problem. Recent experiments involving personal software production found that some programmers with at least five years of experience injected hundreds of defects per thousand lines of code, with a worst case of more than 1,100.

Fundamental lessons from the quality movement imply that the cost of finding and reworking all these defects must be enormous. Most IS professionals do not recognize the scale of the problem, and most IS organizations lack the metrics to quantify it. Even more startling is the improvement that results after three months of using personal metrics: the same programmers who injected hundreds of defects per thousands of lines of code improved their delivered quality by a factor of from 5 to 10.

Moving to the Age of Engineering

Much of legacy management requires IS organizations to move software maintenance and development from craft work into the age of engineering. Mary Shaw of Carnegie Mellon University's Software Engineering Institute has developed a useful model of this transition for several other professional engineering disciplines (see Exhibit 3). In an engineered approach, standards of performance are shared and transferred, patterns of work are made more methodical, and components are interchanged and reused.

Development of an Engineering-Based Discipline

Because this mode of thinking represents a dramatic change in the mind set of the IS organization, opposition to the software engineering perspective can be widespread. IS managers are challenged to foster the IS organization's recognition of the need for change and to reinvent methods and priorities in accordance with the engineering culture.

Managing the Change Process

Most changes to existing systems are processed in a highly inefficient manner; they take approximately four times as long as a change made during new development. IS organizations pressured to respond to business units often attempt the quick fix. Developers routinely enter complex software systems and perform minor fixes (often generating a stream of so-called fix-on-fix errors before completing the job). They then proceed to another task without leaving a record of what they did. The conceptual integrity of the original design, the documentation, and the implemented system itself rapidly
degrade as repeated quick-fix alterations build up like electronic scar tissue. Such systems are inevitably difficult to understand and maintain.

Once again, some organizations are fundamentally rethinking their approach by treating the installed base more like packaged software, which means that change requests are rationalized and managed. These organizations have replaced the quick-fix cycle with an iterative enhancement approach that achieves the following significant benefits (see Exhibit 4):

- They gain a broader perspective of a series of minor changes.
- They resist demand for expensive tinkering that delivers little business benefit.
- They develop a more efficient approach to maintenance because they reduce the 50% of maintenance time that is spent understanding the current application.
- They use advanced techniques such as workshops, business process reviews, and prototyping to ensure that the right problem is being solved.
- They protect the integrity of design and documentation.

**Quick-Fix and Iterative Enhancement Cycles**

Through this approach, business and IS partners exert a far greater level of control and address the real business and technical issue. A formal oversight body or change review board joins business sponsors and process owners with IS portfolio managers. The board maintains a long-range focus (built around the 4R portfolio assessment) and avoids being caught up in the daily routine of change requests. Significant changes obviously require board approval, but even minor adaptations and fixes undergo some less formal scrutiny. Any contemplated change requires the answering of several questions:

- To what degree is the requested change compatible with the original design?
- To what degree is the change necessary?
- How much will the change enhance the system? Does the benefit justify the cost? Is the change compatible with the strategic plan?
- Is the change within budget limits?
- How critical is the change compared to others in the backlog?
- How soon will business payback be realized?
- Are there combinations of changes that generate economies of scale or other synergies?

After these questions are answered, the organization then classifies changes, groups requests by urgency and business benefit, and establishes a review process to schedule changes by priority, cost-effectiveness, and appropriateness for overall legacy strategy. Urgent tasks—such as those associated with a system failure—are performed more
quickly when the system is well documented and skilled people are available. Minor changes with no strong business case are grouped and implemented in a planned release to optimize the maintenance process.

Once a regular schedule of updates and releases is established, business users rely on organized batches of changes, plan the introduction of change into the business, and see the portfolio as a business asset from a long-term rather than a moment-to-moment perspective. Reducing the time IS personnel spend each day on possibly redundant or ill-advised system changes frees them for more types of work.

**The Case for Metrics**

What would business be like without a profit-and-loss statement? What would baseball be like without earned run averages and on-base percentages? For most organizations, the lack of legacy metrics is only partially a matter of technical and methodological issues; it may be a prototypical example of the challenge posed by organizational culture.

Although many organizations thus find it difficult to maintain a history of their projects, estimates, and actuals, other organizations do commit to measurement programs and reap dividends. The results of these programs are striking. For example, a recent report on 500 IS organizations found that between 25% and 30% (even up to 60%) of effort in the few organizations that succeeded in instituting a metrics program is focused on corrective actions on an installed system (in classic quality terms, this is pure rework).

A growing body of support for metrics is originating outside the IS organization in the product and embedded software communities within the business. These outside customers will not return for more poor-quality software. Thus the issue of metrics is best understood as being fundamentally a cultural conflict and a managerial challenge within IS. The main issue is that as long as solid knowledge about past performance is impossible to find, reliable assessments of future performance remain out of reach. As the following sections indicate, many categories of legacy activity lend themselves to accurate measurement.

**Product Measurements.**

How many lines of code were generated, changed, reengineered? How many function points were delivered or modified? How many defects were embedded along the way and how many errors were discovered before delivery? After delivery?

How well do the products satisfy customer needs? How well does a given system add value to or enable a business process?

**Process Measurements.**

How many hours were devoted to which projects in the past 12 months? How many of those were direct, indirect, and managerial? What funds were expended where and when? How much time was spent adding value to the application? What were the cycle times? When was testing performed and how effective was it? What kind of work was being performed?

The following four categories have been suggested for maintenance work: corrective (i.e., fixing mistakes), adaptive (i.e., keeping up with external regulations and changes in technology), perfective (i.e., changing user requirements or performance), and preventive (i.e., enhancing maintainability and reliability).
Organizational Performance.

How well does the IS organization compare with industry standards such as the Software Engineering Institute maturity model? How does the IS organization satisfy its customers? How well is IS delivering value and responding to needs?

Once a system of metrics is in place, managing users, legacy systems, and IS people becomes more straightforward. Without metrics, management remains a matter of educated guesswork. Some of America's most admired corporations—such as Motorola and Hewlett Packard—have recognized the urgency of the issue and are driving metrics throughout their organizations.

Recommended Course of Action

What has come to be known as the legacy problem is not only a matter of today's inheritance of yesterday's assets. Tomorrow's legacy must be considered as well. If today's IS organizations use the same procurement and maintenance processes, handle the same customers using the same change control and architectural approaches—and then have to cope with four or five times as many technology choices—they cannot expect anything better five years down the road. One might even argue that the legacy situation of five years from now will be an order of magnitude worse than it is today.

As with any major reengineering effort, issues of politics and culture are significant. Traditional wisdom on integration has left a legacy of complexity. This inheritance must be managed in a world where the IS function no longer has the monopoly franchise on rapidly changing information technology. These challenges create the need for IS managers to both reengineer IS itself and provide the leadership and understanding critical to rethinking perspectives and driving the change.

Legacy is not an episodic, one-time hangover from the days of the mainframe. Legacy is the ongoing challenge of leveraging evolving IS assets in the era of hybrid computing.

Author Biographies

Christopher Slee
Christopher Slee is a senior partner in the Consulting and Systems Integration Unit of Waltham MA-based Computer Sciences Corp. (CSC).

Malcolm Slovin
Malcolm Slovin is a principal in the Consulting and Systems Integration Unit of CSC.

© Computer Sciences Corp.
Figure Out the Code 50%
Test 25%
Make Change 5%
Document 10%
Plan Change 10%

Source: GUIDE Study on Legacy Maintenance