Rational® software

Tivoli® software

# Moving Beyond IT Optimization

## Guidance for bridging the business, development, and operations divide in IT

**Contents**

## Introduction

How can IT better serve the business? This is the fundamental question CIOs and other IT managers ask themselves when they consider their organizations' abilities to rapidly, efficiently, and cost effectively deliver support for the strategic directions identified by executive leadership. This whitepaper introduces readers to a process and product framework that brings accountability and control to all phases of the IT lifecycle, ensuring optimal alignment of IT with the business. This approach essentially requires that you run IT like a business, making it responsible for delivering ever-improving ROI across the three classic metrics of cost, quality, and time.

We will discuss a process that embraces these three drivers of value, and break down how CIOs and other IT managers can take a measured, manageable approach to adopting it. Through this commitment, managers can break down unseen barriers to IT optimization, enabling themselves to transform their organizations into truly On Demand enterprises. The key? Breaking down the walls that separate development and IT operations.

## Development, IT operations, and process

If data processing professionals of the mid-1970s could travel in time to today, they would be amazed by the sheer scale of technological advance we take for granted. Data processing has been transformed into information technology and has gone from being a back-room support operation to the backbone—and in some cases, the entire body—of modern enterprises. Hardware has continually improved through 20-odd Moore's law doublings; software development has evolved from flowcharts and punch-card editing to today's complex and highly tooled endeavor.

One thing these time-traveling professionals would recognize is our organizational structure, which has hardly changed. Today's separation between development and IT operations remains as it was when it first emerged from out of the laboratory and into the commercial world. Our time travelers might be tempted to ask: With everything else that has changed, does this development/operations split still make sense? Have businesses reviewed this organizational division and still found it to be most beneficial? Or have these folks been too busy with technological progress to go back and ask?

We might also ask ourselves the same questions: Does this organizational structure continue to make sense? Do modern teams still benefit from the division of labor along the development versus operations lines? If it is the best structure, does it have some drawbacks nonetheless? Can we do something to address these drawbacks?

## The development/IT operations divide

At first glance the answers to these questions are simple: Yes, the organizational separation of development and IT operations does make sense and it does serve an economic purpose. Tradition has played such a minor role in IT in the past 30 years—indeed, IT has often been a lightening rod for change—that it is unlikely this split has survived for sentimental reasons. While IT has seen no shortage of proposals for change over the years, only a small number actually make it into common practice. In fact, in all this time, no one has seriously questioned whether, for anything other than the smallest IT organizations, the development/operations split makes sense.

However, it does make sense because development and IT operations are distinct undertakings. Not only do they undertake different work with different tools, processes, and cultures, they are also two distinct activities from an economic perspective. Software development focuses on creating new value; IT operations focus on the optimization of costs.

Given these wide distinctions, it pays to maintain the division between development and operations organizations. As with many undertakings, specialization underpins organizational effectiveness, making the separation of development and operations specialties pay off.

Specialization, however, only works up to the point where we uncouple work activities. Where we couple these activities, specialization can lead to sub-optimal results: processes overlap and redundancy occurs or, alternatively, processes don't coincide, creating gaps. Development and IT operations meet in a place of coupled activities and sub-optimal results.

### The wall—and how it's changing

Where development and IT operations meet has long been described as a wall. The process interface is commonly characterized as "development finishes its testing and then throws it over the wall to operations." For many enterprises, this accurately describes how these two organizations and their associated processes joined—or rather didn't join.

Historically, as earlier noted, this hasn't been a problem. With little linkage between processes, the benefits of separation far outweighed the costs of any overlaps or gaps resulting from the wall.

Today, however, the assumption that development and IT operations don't link is, for many enterprises, no longer true. And with coupling between processes in place, the assumption that teams can continue to "throw it over the wall" no longer holds. We won't argue that, organizationally, we should combine development and IT operations. However, clearly some reengineering of both processes and their interfaces to each other is much overdue.

What is driving this change? For most enterprises, there are three major drivers:

1. *The emergence of complex composite applications*
2. *The decreasing time between application updates*
3. *The rapid rate of new technology adoption*

**Composite applications**
Application platforms enabling technologies, such as J2EE and .NET, have provided development organizations with a comprehensive toolkit for creating new applications of unprecedented complexity. Certainly application platforms enable us to develop new applications more rapidly, but more importantly, they simplify connecting applications. Today's applications are often described as composite; that is, they are composed from a wealth of existing application assets.

Such composite applications are truly powerful: They enable businesses to transform existing processes without having to rewrite large numbers of existing applications. That said, they tend to couple the development and IT operation processes. Because composite applications link many existing applications together in a complex way, many problems may not emerge until the application runs in a live production environment. This is not a problem with testing—good tools are available to ensure new applications work correctly—but rather it's a question of how the new application impacts the existing environment.

We can see this coupling in the emerging application support teams of many enterprises. These groups, often formed from members of a prior development group, ensure that application changes work in the production environment. Rather than throwing composite applications over the wall, these teams nurse them through an extensive acceptance testing period.

**Shorter lifecycle times**

New application technologies have also shortened the development lifecycle. We no longer measure application release timescales in terms of months; modern development tools and methodologies help shorten delivery times to the point that production issues with the previous release are still being resolved.

This is not just a technology issue. Business pressures demand we deliver new IT functions as early as possible; customer expectations are now set in terms of weeks if not days. Rushing to production with the intent to eliminate remaining problems as they occur in live systems is often a valid business judgment; the result, however, is that software quality becomes an IT operations issue and needs to be factored into the service-level management process.

This clearly couples development and IT operation processes. Rather than a wall between the two departments, we have a rapidly moving conveyor belt.

**Technological change**

Technology also drives change. First, with reduced development cycles, we don't have the time to understand new technologies and work through their associated issues before applications go into production. Second, today's technology is such that applications are no longer simple discrete binaries compiled from standalone source code. A composite application is not only made up of code but also of a complex package that includes middleware customization, stored procedures, workflow definitions, and the like. We no longer throw code over the wall to deploy an application; instead, we update a complex and potentially fragile production environment.

### The impact on business and IT management

Whether or not coupling exists between development and IT operations is not an issue from a business perspective. End users have not demanded that we remove the development/operations wall; they have simply demanded that we deliver higher quality, cheaper applications in less time—and, when the end user is the business itself, at (or below!) budget.

The fact is, of course, that the divide does have a negative impact on business in terms of cost, quality, and delivery time.

### Cost

We increase costs because we overlap the processes in development and operations. Take the deployment of applications, for example. From an operations perspective, deployment into production is typically a well-understood, controlled activity with some level of tooling, process management, and quality assurance in place. Applications are, however, often deployed—in fact, much more frequently deployed—into test environments. This process, for most IT organizations, is much less controlled, process-oriented, or automated than its operational counterpart.

So for application deployment—and many other activity areas—there are costly overlaps between development and operations groups.

### Quality

From a quality perspective, however, the gaps rather than the overlaps are the issue. In the divided world, development's goal is to deliver applications not to customers or end users but to operations. Operations then takes the delivered application and makes it work in the production environment. Although performance and stress testing are important, final scalability and real-life usage scenarios provide the ultimate test. Until IT organizations redefine development's goal as delivering applications to customers and end users, that final testing will remain a distinct, operationally separate process from the rest of development. Not only is this costly, but we clearly lose quality across process, organizational, and tools boundaries.

### Timeliness

Finally, the biggest business impact of the organization/process divide is on time to delivery. The mismatch between processes adds friction to the overall lifecycle and slows down the overall IT lifecycle. At a minimum, the organizational priorities and work schedules of development and operations—due in large part to their differing, if not conflicting, goals—creates lifecycle bottlenecks. Beyond that again, tools and process mismatches make the situation worse.

Wherever the specific problems lie in individual organizations, we cannot underestimate the negative impact on business. The common business experience today is not one of throwing a new application over the wall to operations, which quietly makes it work in the real world; rather it is one of repeated shuffling back and forth between operations and development to iron out application problems and, concurrently, address the customer need to rapidly re-iterate the IT lifecycle—in order to add capability in a highly competitive business environment.

### Overcoming the divide

The challenge, therefore, is to overcome the barriers between development and IT operations organizations to improve effectiveness and productivity across the whole IT lifecycle. The key to being successful is achieving a fine balance between actually delivering changes and minimizing disruption.

At the disruptive end, we replace both the development and operations environments into a consolidated whole. This might work in limited cases where a single technology approach is possible and the application scope is narrow. For most enterprises, however, such approaches are either limiting or not scalable. In the enterprise world, technologies, such as J2EE, start out by trying to be closed solutions but inevitably become open ended as the business demand for highly functional, complex and mission critical environments grows.

We are left then with the challenge of managing a complex environment across the whole IT lifecycle. The challenge is not in building a comprehensive application platform but rather in building a comprehensive IT lifecycle management approach. This goes far beyond trying to retrofit existing processes, such as test into new arenas. This challenge calls for the definition of new common lifecycle processes and the implementation of processes, organization, and tooling to support this new process.

We have focused on the gaps and overlaps between organizations but not their commonalities. By establishing common organizational goals, we can build common process models that work in favor of both organizations, achieve the goal of lifecycle alignment, and yet minimize disruption to ongoing workloads.

This last point, minimizing disruption, is very important. Process integration is important to the extent that organizations are coupled. As discussed earlier, coupling occurs where rapid, iterative development of new and updated applications takes place. Where things are more stable—for example, in legacy application management across the lifecycle—a wholesale root and branch process reengineering approach is likely to be more disruptive than beneficial.

Therefore, the bottom line in IT lifecycle management is pragmatism. Improving the lifecycle comes down to implementing workable solutions that bridge the development/operations divide, while leveraging existing tools, technologies, and to a large extent, organizations.

## Common goals and lifecycle initiatives

As noted above, to deliver IT lifecycle management solutions, we start by going back to what the development and operations organizations share as common goals. Although development is a driver for change and competitive advantage and IT operations are a driver for organization productivity and resilience, they both share the following three goals:

- *Better alignment with business objectives*
- *Ongoing quality improvement*
- *Improved process agility*

In each case, the business outcome is the same but the organizational imperative differs. In terms of business alignment, for example, development focuses on building the right application functionality and operations focuses on delivering improved service at reduced cost. Likewise in quality improvement, the development focus for quality is, typically, on reducing defects; whereas the operational focus is on meeting availability and service-level targets. Finally in agility, development focuses on flexibility and reuse; whereas the operational focus is, typically, on optimizing resources.

Simply joining processes, therefore, is not the approach to take. Alignment needs to come from identifying crucial work threads that cross the lifecycle and then applying the right processes and tools to facilitate these threads. These crucial threads, or IT imperatives, consist of sub-processes from both development and operational activities that help move some aspect of the application across the entire lifecycle—from planning through development and into the ongoing production environment.

Consequently, rather than replacing it with a wholesale process or adopting a new (and limiting) technology, we should approach IT lifecycle management through focused execution of these IT imperatives, aiming to deliver better work processes across the threads where development and operations intersect. But what are these imperatives and how best to ensure successful execution?

## IT lifecycle solutions—The Rational®/Tivoli® connection

With the assistance of IBM's Global Services organization, Rational and Tivoli have worked together to understand the common development/operations work threads and to design a set of initiatives that customers can undertake to improve process and results across the IT lifecycle. This list of IT imperatives is not exhaustive by any means but, based on customer input, aims to address the areas of biggest concern—and potential benefit—in the area of lifecycle integration and process improvement.

The remainder of this paper will cover four of these IT imperatives:

- *Delivering a common process for IT governance, portfolio management, and IT investment planning*
- *Improving application functional quality and time to application acceptance in production*
- *Improving application performance and the reduction of application reengineering for performance reasons*
- *Acceleration and auditability of component deployment into both development, test, and operations production environments*

For each imperative, we have identified common work threads and tooling recommendations. In each case, you must maintain the existing development and operational processes and tooling. Successful execution builds upon existing processes to bridge both organizations. In this way, you can make IT lifecycle management improvements in an evolving, iterative fashion while maintaining existing investments in process and technology. So, for example, an organization using the Rational Unified Process in development and IT Infrastructure Library (ITIL®) service management guidance in operations can continue to use and evolve these practices while implementing lifecycle solutions.

These IT lifecycle imperatives are meant to be flexible and be adapted to specific organizational needs. Rather than regarding them as prescriptive, think of them as descriptive best practices. As part of these best practices, IBM has developed particular product integrations and workflows. Again, these are not prescriptive but can be used as the basis for implementing a solution that best fits a particular organization's needs.

**The following four sections describe each IT imperative in more detail.**

## Governing IT across the lifecycle

Although IT across the lifecycle has always been actively managed, implementing a formal IT governance strategy—where we treat IT as more than a cost center—is becoming a priority for many enterprises. Driven either by the realization that formal controls will ensure better management decisions or from the need to respond to regulatory pressures or requirements, IT governance provides a framework for overall IT strategic management, investment analysis, and linking of IT activities to business decisions.

Although IT governance has only recently emerged as a formal standalone discipline, many of the underlying activities have been in place for some time. As you might expect, these activities are split across the development/operations divide.

In the development space, governance is often seen as an investment or portfolio management issue. IBM's "Your Turn: The Global CEO Study 2004"[1] established that 85% of interviewed CEOs identified revenue growth as their primary objective, so deciding where to invest, and subsequently measuring return on investment, is a key piece of good corporate governance. Balancing and tracking investments across the application and business portfolio has thus become a critical process for many organizations.

From the operations perspective, governance is often expressed as the need to align service delivery with business needs in order to improve process effectiveness and deliver the best service at the lowest cost. Service level, availability, and capacity management combine to provide a core set of processes by which you define, measure, and improve service.

1. http://www-1.ibm.com/services/ondemand/business/global_ceo_study_2004.html

If you don't require linkage between development and IT operations, you don't need to unify the two governance models; you can keep the application portfolio and service delivery management processes separate. In some cases, this kind of strong separation exists; for example, where an enterprise outsources a third-party hosting business to run its production environment. In most organizations, however, linking governance activities is desirable.

At the heart of governance is making portfolio investment decisions. A typical decision might be characterized as: Shall I build application A or application B to get the best return on investment? In an IT lifecycle management context, however, you must supplement this with questions, such as: Would I be better off improving the service for application A or building a new application B? Being in a position to make these decisions enables IT management to make much better governance decisions.

To do this requires bringing together the process and tooling for two previously distinct processes—project portfolio and service-level management—into a common solution. As the critical issue is management decision making, the key requirement is to provide common management reports for project and service-level status across the lifecycle.

IBM® Rational Portfolio Manager and IBM Rational Team Unifying Platform® together with IBM Tivoli Business Service Manager and IBM Tivoli Service Level Advisor provide the ideal platform on which to build this integrated process. Rational Portfolio Manager provides visibility into business and financial metrics for IT projects across the portfolio, delivering insight into project resource availability and utilization. On the service delivery side, Tivoli Service Level Advisor and Tivoli Business Service Manager together provide similar business-facing metrics for IT service delivery.

By incorporating service delivery performance indicators into the overall IT portfolio scorecard, IT managers along with line of business and customer management can get a complete picture of the overall functioning of IT as a combined application and service delivery organization. This integrated perspective paves the way toward making balanced governance decisions and maximizes the ability to make the most effective investments across the lifecycle.

## Ensuring functional quality of applications

Seeking to improve application quality is, clearly, nothing new and comprises a large part of the ongoing evolution of software development practices and technologies. For most organizations, this process improvement and the day-to-day assurance of functional quality in applications are development responsibilities; what happens after an application is turned over to production consists largely of shaking out the final defects. Essentially the working assumption is that debugging has finished when the application is moved to production.

In the past, this approach made sense. With long timescales and relatively small amounts of middleware content, IT organizations could afford to finish the development cycle with a final systems test and then pass the code (typically as binaries) to IT operations to put into production and iron out any problems.

Today, the application lifecycle is typically far more rapid, the tooling more complex, and applications not only consist of binaries (or, more likely, class libraries) but also large amounts of middleware customization and programming. In this environment, the lines between development and operations responsibilities become blurred. We cannot catch all defects in development and we need to bring production management to applications as early as possible—preferably in the test environment—in order to catch problems early.

Adding to these problems is the issue of scalability. Testing to ensure that an application will scale in the production environment has always been a difficult process, where we cannot know all potential issues in advance. With today's Web-based, customer-facing applications, even the overall scaling factor at deployment becomes an unknown. Many organizations have experienced scalability problems in applications that become used much more heavily than anticipated.

Here the divide between applications and operations becomes a critical bottleneck. As the traditional model passes only application assets across the wall, we effectively lose all potential intelligence into key scalability and quality stress points (or, at least, they are hidden behind the wall!). Improving quality of applications across the lifecycle requires us to gather this intelligence and apply it to production quality and scalability efforts.

Quality cannot be tested into an application: It starts with getting the requirements right and goes all the way through to ongoing production monitoring and control. Ensuring quality is not just about good design and build practices; it also involves capturing critical architectural, design, and construction information and making that available to operations.

In essence, the imperative is to expand our definition of functional quality beyond the idea of software defects to include how the application functions in the operational environment. In setting up this operational monitoring, the development organization is likely to have the best insight into where—from an architecture, design, or construction perspective—the failures are likely to occur. These insights should be the basis for operational control.

To do this, you use the development process's requirements, architecture, design, build, and test phases to drive operational requirements and management implementation. You can use the Rational toolset at each development phase. As applications move through this process, IBM Rational RequisitePro® can collect management requirements and IT operations can then use them later to build in the management required to ensure functioning, operational applications.

Operational quality is no longer simply about resource management. In the past, the relationships between application components and OS and middleware were fairly simple and managing operations was a systems-based activity, focused typically on tuning or configuring systems components or sub-systems. Today, we need to supplement systems management needs with in-depth analysis in the application's real-time production systems. Modern composite applications—applications using a technology, such as J2EE, to tie together (to compose) multiple services, databases, and existing applications—need diagnostic tools that go beyond simple management of server and middleware resources.

IBM WebSphere® Studio Application Monitor is such a product. Taking problem detection, isolation, and diagnosis back to the application itself enables you to address problems within it. This then ties back into the development process, driving requirements for fixes, maintenance, or upgrades to address problems according to their severity. Using a combination of IBM software tools to allow the development and operations processes to work together, IBM delivers a closed loop for ensuring application functional quality across the IT lifecycle.

### Ensuring application and system performance

In contrast with ensuring functional quality, the operations side of IT has traditionally taken the lead in ensuring the performance of both applications and systems. Certainly application groups had guidelines for avoiding the worst mistakes in delivering poorly performing applications, but a larger fraction of the performance improvement opportunity was typically available via tuning systems.

Today, the opposite is true. For composite applications, the major performance tuning opportunity is in the application itself—not only because it is easier to modify today's object-based, componentized applications but also because the applications themselves drive multiple service and middleware components in each transaction. Finding better ways to do so can lead to significant performance improvements.

Ensuring system performance, however, can extend beyond simply increasing the scope of an operations discipline; you can extend it to the whole IT lifecycle. To start, consider customer requirements. At the project's outset, customers typically have an outcome in mind: They need a certain set of functions and business capabilities. Inherent in these requirements are an often unstated set of assumptions about service delivery—the desired availability, resilience, and performance of the service delivered to the end user.

Therefore, the start of our integrated process calls for the collection of service-level management requirements just like any other application feature or aspect the customer desires. Standard requirements management tools, such as IBM Rational RequisitePro, can collect such requirements. At a later date, you may need to negotiate or confirm a service-level agreement with the customer; however, you can do that from the basis of taking the requirements back to the customer and building the detailed terms and conditions from there.

As you refine the service requirements, IT operations can use them as input into developing the service-level monitoring and reporting needed to manage the application. This avoids the typical reengineering of service-level requirements that happens in many projects close to their release. It also enables a formal linkage early in the lifecycle between the customer and any service-level management process that operations has in place, say through an ITIL implementation. Finally, it provides the opportunity for both operations and development to come together early in the development lifecycle in order to review requirements and uncover any issues that might impact both development and operational aspects of having the application meet its service levels.

Once an application is in production—and service-level monitoring and management is in place with Tivoli's service delivery products, such as IBM Tivoli Service Level Advisor, IBM Tivoli Business Service Manager, and IBM Tivoli Enterprise Console—you can establish continuous monitoring and feed it back into the service-level management review process.

You can also implement these tools in test environments, along with IBM Rational Performance Tester, to benchmark new or modified applications and to ensure the correct operational management definitions (thresholds, critical alert definitions, etc.) are passed into the production environment. Further, by linking application diagnosis for services failing to meet service levels in IBM WebSphere Studio Application Monitor and back into the development process through IBM Rational Application Developer, you can implement an effective closed loop define-monitor-analyze-correct-deploy process across the IT lifecycle.

An IT lifecycle management process that links requirements to service-level management to application diagnosis and repair ensures that customer and business requirements are met and service levels are maintained. It also eliminates the gap between development and operations that may otherwise hinder an application meeting its objectives and expected ROI.

**Accelerating deployment**
The fourth IT lifecycle management initiative addresses the issue of the wall between development and operations directly. That wall is, after all, the place where development traditionally signed off on a new or upgraded application and passed it to operations to deploy into production.

Historically, this wall was ideal; it enabled the two organizations to work in a decoupled fashion to their own goals, objectives, and timescales. Today, as discussed, this decoupling is no longer desirable due to both technical and business pressures; its replacement with integrated processes is also an emerging IT imperative. From a process perspective, nothing is closer to the wall in the IT lifecycle than the application deployment process. Therefore, you should expect that accelerating deployment is a key IT lifecycle management initiative.

There are two major pressures on the deployment process today. First is the ever-increasing complexity of the application component manifest that makes up an application change or release. The contents of application releases have gone from being simple library changes (typically, binary runtime files) to complex aggregations of objects, libraries, queries, definitions, rules, workflows, and so on, depending on the technologies utilized. Simply moving all these components into production in a timely manner poses a great challenge.

The second pressure comes out of the compressed nature of the development lifecycle and the need to iterate through testing cycles ever more rapidly. Deployment was historically done at the development/operations wall and developers and testers used their own, often manual, approaches to create their own environments. Not only does increasing application complexity lead to an increase in the chances of getting the test deployment wrong—and hence, essentially, of testing the wrong application—but the iterative nature of today's development process demands an automated approach to avoid the test-to-deployment handoff from becoming a major bottleneck.

To provide an integrated solution, you can combine IBM Rational ClearCase® and IBM Rational ClearQuest® with IBM Tivoli Configuration Manager. You can use Rational ClearCase to provide build information directly to Tivoli Configuration Manager to automate builds and their deployment. For remote deployment, you can also use IBM Tivoli Configuration Manager to extend the reach of automation. For customers seeking highly flexible environments, you can integrate IBM Tivoli Intelligent Orchestrator to provide resource deployment in addition to application deployment.

By using a common deployment environment for both testing and production, you can accelerate development lifecycles and production deployment errors, or you can largely eliminate the possibility of tests being certified against an incorrect build. For many organizations, the IT Lifecycle Accelerate Deployment imperative is a beneficial and practical starting point for merging the development and operations processes across the IT lifecycle.

**Getting started**

In each of the four IT lifecycle imperatives, we have seen how we can improve the overall application development and delivery process while minimizing the impacts to existing processes and investments. To the extent that development and operations activities become linked, and therefore, to the extent that common work practices and tools need to be shared, IT lifecycle management provides for the integration you need to deliver more effective IT processes and business value.

Often the most difficult part of any project is getting started. This especially applies to process improvement projects. Such projects are also prone to suffering scope creep and to growing in size and ambition until they become unmanageable. To avoid this, our experience suggests the following best practices in implementing IT lifecycle management initiatives:

- *Pick one project with a known problem*
- *Get management sponsorship*
- *Enact good project management—especially through defined outcomes*
- *Implement, measure, then iterate*

Some of these points are, of course, applicable to any type of project. The first point, however, is worth some elaboration.

All process change is disruptive and subject to resistance. This is particularly true in IT lifecycle management projects where, organizationally, one group can easily suggest the problem is with the other group! Management fiat works in such cases—and clearly strong sponsorship is a good idea—but beyond that, finding a common, known problem is a great starting point. In consulting and researching IT lifecycle management with IBM customers, IT operations personnel and management understood better than their development counterparts that the general issue of the wall being a problem was worth attention. This is generally an awareness issue driven by operation's downstream position in the lifecycle. However, to be successful, both organizations need to recognize a common pain that needs to be tackled.

These pains are discussed in summary in the initiative descriptions above. For most organizations, each one will be an issue to some extent. A useful place to start an IT lifecycle management project is by reviewing each pain, identifying the area of greatest potential impact, and then picking a particular project or application area as a pilot. Once one initiative is in place, measure the benefits, re-analyze priorities, and pick a next target.

The bottom line is that focusing on these IT lifecycle imperatives introduces a means for overcoming the challenges of IT lifecycle management—simple yet effective ways to get real benefits quickly and form the basis upon which to build a sustained process improvement project.

## Summary

We started with the question: How can IT better serve the business? A blanket answer to this question is difficult to provide but it becomes clear that coupling development and operations processes driven by today's business and technological maturation has exposed new challenges. In the past, teams could rely on the wall between development and operations to be an acceptable, even desirable, state of affairs. Today, however, this wall exposes IT to cost, quality, and responsiveness issues for most organizations and needs to be addressed.

While there are no quick technological fixes, addressing these issues does not involve a major upheaval. The approaches to successful IT lifecycle management outlined in this paper provide pragmatic guidance for tying development and operations together, streamlining workflows, eliminating duplication of effort, and removing common points of failure in cross-process, cross-organizational communications.

By breaking down key internal barriers, we can meet the challenges of today's technology, application, and business environments and accrue the benefits to both development and operations and, most importantly, customers and end users.

G507-0953-00