



## Reduce IT Costs and Complexity with Effective Application Problem Management

By Herb VanHook, Vice President of Corporate Strategy, BMC Software

# Table of Contents

- EXECUTIVE SUMMARY** ..... 1
  
- COST OF DEFECTS** ..... 2
  - > Complex Applications.....2
  - > Organizational Changes .....2
  
- CONTROLLING PROCESS DEFECTS** ..... 3
  - > Detection.....3
  - > Isolation.....3
  - > Resolution .....4
  
- PROBLEM MANAGEMENT** ..... 4
  - > Comprehensive Application Problem Management .....4
  - > Benefits of Automating Problem Management Through Problem Detection, Isolation, and Resolution.....6

## Executive Summary

Most business processes are automated by software applications. As these software applications become indispensable, you might think that they would likewise become more dependable. Instead, system complexities and interdependencies are making enterprise application problems inevitable, intractable, and more elusive. For example, these problems can cause an airline to accidentally cancel flights, a bank to deny banking services to customers, a retailer to inadvertently put a halt on customer orders, and so on. The worst-case scenarios can be overwhelming.

Given the time, cost, and impact caused by software defects in recent years, you might expect some progress in the state-of-the-art techniques for finding and resolving defects. While advanced application design and new development and integration technologies have emerged, the pure effort involved in root-cause analysis for most organizations is stuck in the past. The processes remain labor intensive. Voluminous amounts of information must be gathered to track down, isolate, and resolve problems. Often, multiple attempts are made to recreate the problem. Detailed analysis of the problem data requires teams of specialists. In addition, many of the multitiered and “loosely coupled” applications aggravate the exercise of isolating and solving problems in production systems.

Frequent group meetings attempt to tackle problem isolation. However, “all hands on deck” exercises disrupt process-based operations and project-based development. Finger pointing and chasing down dead ends waste more time. Organizations need a cleaner and more direct approach to overall problem management. Such a method is essential to ensuring continued system availability and limiting the impact to business operations.

This paper presents recommendations for comprehensive problem management and describes how IT organizations can address problems related to performance and availability, while reducing costs.

## Cost of Defects

The cost of defects is far greater than you might expect. The National Institute of Standards and Testing (NIST), in a study of software errors, determined that defects cost the U.S. economy about \$60 billion annually.<sup>1</sup> This cost mainly comes from the impact on business availability and performance. It includes the labor cost to find and fix these problems. The report also finds that 80 percent of the time spent in managing an application lifecycle is spent fixing defects.

Not surprisingly, the study also indicates the cost of fixing problems rises dramatically, later in the application lifecycle. Fixing a defect in production is six times more costly than fixing one during unit testing.

IT groups need a better way to tackle problems. In a recent report, Fawcett Publishing discovered that only 9 percent of IT professionals surveyed described themselves as very satisfied with their diagnostic software and processes. And at least one-third reported that they rely mostly on manual and homegrown approaches, such as “eyeballing” logs, spreadsheets, or internally written scripts.<sup>2</sup> The current processes result in a long “shakedown cycle” and exacerbate the business impact that problems cause: significant revenue and productivity losses; customer satisfaction issues; and the resource drag on the development team as complex, intermittent problems needlessly escalate.

### Complex Applications

It has been a long time since the “good old days” of monolithic applications, where all code existed as a single executable chunk — with calls to make sure system routines cannot be broken. Client/server application architectures, and later Web-based architectures, caused code to be broken into increasingly smaller pieces. The introduction of application servers, integration brokers, and Web servers as software infrastructure has also added to the complexity.

Recently, Service-Oriented Architectures (SOAs) have taken hold. SOA promotes loosely coupled models of applications — often with Web Service invocations passing metadata documents. These layers of abstraction have made “getting

to the root of the problem” that much harder. Even more application infrastructure components (e.g., Enterprise Service Buses or ESBs) are being added to the complexity stack.

Code failures, performance bottlenecks, errors in data translation, and similar issues are difficult to track down. SOA models cause a division of labor, where the application path (or transaction path) comes from code components written by many hands. Often, it’s difficult to find the singular expert that understands the complete code execution path.

### Organizational Changes

These new application architectures and application infrastructures are driving organizational changes. Organizations are establishing dedicated application support groups that serve as a bridge between development and operations.

IT operations groups have long been responsible for keeping production running, and have evolved into specialists in operating systems, subsystems, databases, and such. But these teams have come under cost constraints in recent years. The new software infrastructures challenge IT operations to keep up with specialist skills.

New code development methods push applications “over the wall” into production faster than ever. Rates of business change are increasing, driving system changes. Delivery demands are squeezing testing cycles. Complex application architectures make comprehensive testing difficult. All too often, testing is not performed to a determined quality level, but instead, is executed for a fixed period because it was called for in the project plan.

Once an application reaches production, its care and feeding becomes paramount. Dedicated application-specific teams have emerged to manage changes and deal with problems. These teams become level 1, 2, and often level 3 application support personnel. They develop the skills to understand the application infrastructure environment. Hence, they become specialists working with the diagnostic and monitoring tools that once would have been the domain of IT operations teams.

Requirements Gathering and Design	Coding / Unit Test	Integration and System Test	Beta Testing	Production
1X	5X	10X	15X	30X

Source: NIST Publication #2002-10. X is a normalized unit of cost and can be expressed in terms of person-hours, dollars, etc.

Figure 1. The cost of fixing production defects

## Controlling Process Defects

In controlling process defects, the fundamental steps are detection, isolation, and resolution. Each of these steps must be addressed comprehensively to successfully and quickly fix defects.

### Detection

Much of problem detection still is based on external input — someone contacts the help desk with a problem: an application isn't working; the output looks bad; the transaction failed. Monitoring infrastructure availability and performance is commonplace, but transaction and application monitoring is less common. The rate of change in applications can complicate monitoring efforts. As applications and transactions change, the detection mechanisms may need to be updated.

Companies that lead technology best practices tackle this problem proactively with an approach that involves monitoring the basic infrastructure for availability, performance, and throughput. This approach also requires similarly monitoring the application infrastructure. Additionally, monitoring transactions synthetically (using simulated users) to see a round-trip view of transactions is often necessary. This process can give early warning of problems before users see them. Another proactive approach involves watching real transaction performance. It includes using compliance tools for data validation, and consolidating in one place all events related to problem watching.

These detection methods must be coordinated with any help desk or service desk processes. Correlation of user-reported problems and system-detected problems must occur. Efficient triage of problem routing is also necessary. Solid process and workflow models help dictate who gets "first look" at each type of problem event.

### Isolation

Because most applications have multiple components — different people and organizations responsible for various aspects — once a problem is detected it must be isolated. The system, area, or component of the application that is causing the problem must be determined, so that the analysis effort can be focused accordingly (e.g., if the problem had nothing to do with the database, the DBA team does not need to waste any time or process cycles on the issue).

Making these determinations is a major data-gathering exercise that involves collecting system logs, configuration files, change histories, data dumps, and other information. Depending on the severity of the problem, groups of people responsible for managing the problem come together in a "best minds in the room" exercise. Isolation is done to determine who gets to leave the room. This work can expand to include end-user interviews and even costly onsite trips.

Isolation usually proceeds from coarse-grained to fine-grained approaches. The team brings up questions, such as: *In which system does the problem reside? Is it network related or database related? What changed recently?* This thinking proceeds in steps, as systems and application analysts hone in on the problem.

If everything works as planned, this focus can resolve the problem to a single line of code, or a single configuration parameter, and so on. All too often, however, that is not what happens. Suggestions for recreating the problem are typically introduced. In fact, much of the information gathered is used only to reproduce the problem environment. Can we make this fail again? And if it fails, can we get enough information about what is going on at the time of the failure?

IT groups spend little time on investigating single failures. If the problem cannot be recreated, or doesn't recur, it is usually forgotten. The chronic problems get the focus. Isolation can sometimes take days or weeks. As each stage of diagnosis requires another round of data gathering and analysis, time stretches on.

This iterative process remains a manual, error-prone, communications-intensive activity that consumes resources. It ties up developers and testers if the problem occurs in preproduction; if found while in production, it ties up developers, operations, and support personnel. Studies show that problem replication efforts, on average, involve three to nine experts and some estimates indicate that problems are recreated nearly five times in attempts to isolate a root cause.<sup>3</sup> Worst case is when the problem cannot be replicated, and thus cannot be resolved.

## Resolution

Problem resolution is a result of true root-cause analysis.

Once a problem is isolated to the proper tier of the application, the trial and error process for the root cause begins. This is an iterative process that involves a good deal of educated guesswork, which does not always guarantee a positive result.

The goal of the resolution phase is to identify the root cause of the problem. At this point, the problem can be explained: a boundary condition was exceeded; a data format was wrong; a line of code was out of place. Sometimes a combination of factors was the root cause.<sup>4</sup> In any case, the root cause of the error was identified. Now a fix must be designed, implemented, and tested.

## Problem Management

In addition to the labor expended to find and fix a problem, process-centric IT shops ensure the management of the overall process is robust. Whether using the IT Infrastructure Library (ITIL<sup>®</sup>) process structure, ISO 20000, or any similar process model, additional steps must be considered for completeness. Problem management process workflows are unique to each organization. However, they typically include:

- > Problem/error identification and recording
- > Problem/error assessment
- > Problem classification — in terms of the impact on the business
- > Problem investigation and diagnosis
- > Recording the error resolution
- > Closing the error and associated problems
- > Capturing detail in knowledgebase for reuse
- > Problem prevention actions
- > Major problem reviews:
  - What was done right?
  - What was done wrong?
  - What could be done better next time?
  - How can we prevent the problem from happening again?

## Comprehensive Application Problem Management

IT groups are looking for better ways to improve application problem management as part of an enterprise strategy for Business Service Management (BSM), the most effective approach for managing IT from the perspective of the business. As stated earlier, the cost to the business is in time, and relates to both people and outages. Proactive and timely problem detection, coupled with automation on the resolution side, is imperative. Application analysts want isolation and diagnostic information gathered automatically. They want to reduce and eliminate the need for recreating problems. And they want tools to accelerate the problem analysis.

When seeking solutions that will deliver a comprehensive problem management vision for production, all aspects of problem detection, isolation, and resolution must be considered.

A complete application problem management solution should provide the following abilities.

### **Monitoring to detect availability, performance, and throughput of problems**

This requires monitoring the application infrastructure and the end-to-end transaction execution flow. To proactively spot problems before they impact end users, a monitoring strategy should include monitoring of synthetic transactions (simulated users).

### **Isolation to quickly categorize the problem and identify the location of the failure**

As any IT veteran will confirm, the symptoms of a software problem rarely reflect the root cause. A single business transaction may trigger a sequence of complex processes, each of which can involve events that occur on dozens of servers. The root cause of the problem could be a software issue, a hardware fault, a configuration issue, or even an end-user mistake. A solution must help in quickly categorizing the problem and isolating it to the correct location (system, application tier, etc.).

### **Full data capture to eliminate the need to recreate problems**

The system must capture the data that would enable root-cause determination, without needing to recreate the problem environment. That data must include user actions, application configuration, access to external resources, application performance, and full code-level execution history. The data must be synchronized to enable easy drill-down throughout the root-cause analysis process.

**Ability to capture data from clients and servers, remotely and locally**

Problems are not limited to one tier or location. They occur everywhere: on servers and workstations; inside and outside the firewall; locally or at remote end-user sites.

The solution must be able to capture root-cause data from all application tiers — clients, Web servers, application servers — as well as from all possible runtime environments, whether local or remote. The solution must also provide the means to gather the data.

**Flexibility to resolve any type of application problem**

An IT organization deals with varied types of problems throughout the application lifecycle. Problems can be categorized as:

- > **Functional problems** — The majority of coding errors, or bugs, are typically uncovered and fixed in QA testing. Some bugs, however, make their way into the production builds, causing applications to produce incorrect results, deny access, or even crash.
- > **Performance problems** — Performance issues are a major pain area for applications. These are commonly a result of non-optimized code, insufficient hardware resources, or incorrect settings within the runtime environment.
- > **Configuration problems** — Distributed applications rely on numerous components, making configuration errors a chronic source of problems. Despite efforts to verify and control production environments, even the most minor discrepancies between development labs, QA labs, and production environments can cause applications to malfunction.
- > **Operations** — On average, 40 percent of unplanned mission-critical application downtime is caused by application failures and another 40 percent is caused by operator errors. Improving change management processes is one of the best investments that enterprises can make, as availability can increase by 25 percent to 35 percent.<sup>5</sup>

**Support for applications running across mainframe, Microsoft Windows, .NET, J2EE, and third-party technologies**

Most IT organizations rely on both J2EE and Microsoft technologies for new application-development capabilities. These organizations also have a significant investment in existing legacy applications. A solution that only supports one of these environments has limited value.

**Role-based views catering to all problem resolution stakeholders**

When hardware (such as a router) fails, the network manager is responsible to replace it. However, when software fails, multiple teams are involved in isolating the problem area, finding the root cause, and delivering a fix. On average, nine people “touch” a problem before it gets resolved.<sup>6</sup>

The solution therefore must provide the appropriate role-based views for all the functions engaged in the problem resolution process. For example, the solution must provide deep visibility into code execution to the application developers. It should enable application support staff to view end-user configuration settings or view end-user actions, and let performance engineers analyze system and application performance.

**Integration with existing systems and processes for production problem resolution including homegrown logging, defect tracking, service management, and help desk processes**

Most IT organizations have implemented processes for reporting and tracking problems. Rather than attempting to define new processes for solving problems, the ideal solution should integrate into the existing processes. This is achieved through integrations with defect-tracking tools and with help desk and systems management tools. These integrations must leverage existing workflows to communicate root-cause data between the different teams engaged in the problem resolution process. Finally, the solution must seamlessly integrate with homegrown logging mechanisms to ensure it is fully utilized by application developers and support engineers that are accustomed to using internal logs as the primary means of problem diagnostics.

**Minimal and dynamically controllable overhead**

To detect application problems quickly and proactively, a transaction monitoring solution must be able to run 24x7 with minimal disruption and overhead in the production environment. A great deal of data must be captured to determine root cause of application problems. The solution must capture that data without putting unacceptable overhead on applications running in the production environment. It must allow the user to balance the level of data capture with the performance overhead. This must be done in runtime, without restarting the application, so that the business process is not disrupted by problem capture activities.

## **Quick and nondisruptive processes for “reactive problem resolution”**

Painless deployment is always desired; however, in problem resolution, it is absolutely key. Although monitoring solutions must run 24x7, the actual problem resolution stage is often a reactive process. The solution, therefore, must support on-demand deployment of capture agents to local or remote environments without disrupting the ongoing operation of the application. The agents should be able to attach to the live application, without forcing a system reboot, so that problems may be recorded while the application continues to run. Finally, the solution must be non-intrusive. It should not require any changes to source code, executables, or runtime environment.

## **Benefits of Automating Problem Management Through Problem Detection, Isolation, and Resolution**

With a comprehensive solution for application problem management, IT staff can simultaneously understand the true end-user experience, see the components involved in end-to-end transaction service delivery, and rapidly identify the root cause so they can find and fix problems that affect critical business transactions — before they hurt critical business services. Advantages can include the ability to:

- > Pinpoint root cause of production application problems, up to 80 percent faster than previous ability
- > Eliminate time-consuming, disruptive processes for gathering information from end users
- > Perform triage more rapidly, to avoid unnecessary escalations to development team
- > Eliminate the need to reproduce the environment or replicate the problem before resolving it
- > Avoid expensive SWAT team deployments
- > Easily meet SLAs
- > Improve customer satisfaction levels by resolving problems faster

Given the critical nature of today’s business applications, an enterprise should develop a comprehensive strategy and consider the deployment of tools and processes to help IT quickly isolate and resolve application problems. This ability will increase performance and availability and reduce risks to the business through effective problem management. BMC Software offers solutions that address these issues. Visit [www.bmc.com](http://www.bmc.com) for more information.

---

### End Notes:

- 1 “The Economic Impacts of Inadequate Infrastructure for Software Testing,” National Institute of Standards and Testing (NIST) Planning Report 02-3, May 2002, Section 6, p. 8-1
- 2 “Troubleshooting Distributed Enterprise Applications,” FTPOne Survey Report, February 2005, p. 2
- 3 “How Visible are IT Problems, Really? The Perception of IT Problems Across the Enterprise,” Dynamic Markets Ltd., Sept. 2004, p. 18
- 4 Ibid
- 5 Gartner Research, Inc., “Best Practices for Continuous Application Availability,” D. Scott and E. Holub, Gartner Data Center Conference, December 2005
- 6 “How Visible are IT Problems, Really? The Perception of IT Problems Across the Enterprise,” Dynamic Markets Ltd., Sept. 2004, p. 18



ACTIVATE BUSINESS WITH THE POWER OF IT.™

### About BMC Software

BMC Software delivers the solutions IT needs to increase business value through better management of technology and IT processes. Our industry-leading Business Service Management solutions help you reduce cost, lower risk of business disruption, and benefit from an IT infrastructure built to support business growth and flexibility. Only BMC provides best practice IT processes, automated technology management, and award-winning BMC® Atrium™ technologies that offer a shared view into how IT services support business priorities. Known for enterprise solutions that span mainframe, distributed systems, and end-user devices, BMC also delivers solutions that address the unique challenges of the mid-sized business. Founded in 1980, BMC has offices worldwide and fiscal 2006 revenues of more than \$1.49 billion. Activate your business with the power of IT. [www.bmc.com](http://www.bmc.com).

### About the Author

Herb VanHook is vice president of corporate strategy at BMC Software, and has held several key positions at META Group (most recently serving as interim president and chief operating officer). VanHook has more than 30 years of experience in information technology, including senior positions at IBM, Computer Associates, and Legent Corporation.

