Realtime publishers

The Definitive Guide™ To

Application Performance Management

sponsored by

Greg Shields

Introduction to Realtime Publishers

by Don Jones, Series Editor

For several years now, Realtime has produced dozens and dozens of high-quality books that just happen to be delivered in electronic format—at no cost to you, the reader. We've made this unique publishing model work through the generous support and cooperation of our sponsors, who agree to bear each book's production expenses for the benefit of our readers.

Although we've always offered our publications to you for free, don't think for a moment that quality is anything less than our top priority. My job is to make sure that our books are as good as—and in most cases better than—any printed book that would cost you $40 or more. Our electronic publishing model offers several advantages over printed books: You receive chapters literally as fast as our authors produce them (hence the "realtime" aspect of our model), and we can update chapters to reflect the latest changes in technology.

I want to point out that our books are by no means paid advertisements or white papers. We're an independent publishing company, and an important aspect of my job is to make sure that our authors are free to voice their expertise and opinions without reservation or restriction. We maintain complete editorial control of our publications, and I'm proud that we've produced so many quality books over the past years.

I want to extend an invitation to visit us at http://nexus.realtimepublishers.com, especially if you've received this publication from a friend or colleague. We have a wide variety of additional books on a range of topics, and you're sure to find something that's of interest to you—and it won't cost you a thing. We hope you'll continue to come to Realtime for your educational needs far into the future.

Until then, enjoy.

Don Jones

Copyright Statement

Chapter 1: What is Application Performance Management?

This Web site is experiencing unexpectedly high volume.
Please try again later.

You've seen these words before. Perhaps you were buying a just-released book or video through an online Web site and it popped up in the middle of checking out. Maybe you were trying to get tickets to that important sporting event or that one-night-only concert. What about when the winter storm of the century hits your airport and thousands of people scramble at once to find a new flight or a hotel room.

Each of these scenarios is strikingly similar to the others. An IT service struggles to keep up with the load of its users, until that load finally overwhelms its capabilities. You, the end consumer, are greeted with a pleasant message that effectively tells you…nothing. You don't know what happened. You don't know the status of the problem or its resolution. You don't even know when that suggested "later" may be for you to try again. So, you—and everyone else—find yourself hitting the Refresh button over and over again impatiently waiting for a better response.

Or, in extreme situations, you stop doing business with that site entirely.

Each of these scenarios is also remarkable in how often they're seen by the end customers of Web and other IT-based services. When they work, the IT services used by businesses are fantastically efficient in servicing customers. Yet when they don't, the result is the online equivalent of a "Closed for Business" sign hanging on the front door.

You might have experienced situations like this in other places. Perhaps the problem isn't in an online e-commerce system. Maybe a similar outage of service happened within an internal business application, the functionality of which is critical to getting your job done. Maybe an underlying IT infrastructure component such as name resolution or the network itself experiences a problem. The result of that low-level issue manifests itself in ways that are seemingly unrelated to the actual problem.

The central problem in all of these situations is an inability to properly manage application performance.

More than Just On vs. Off

If you're an IT professional reading this guide, you've heard these stories many times before. You know about the host of potential problems that an IT infrastructure can and does experience on any particular day. You've experienced the nightmare situation where a critical service goes down and no one can track down exactly why. You've sat in the "war room" where highly-skilled individuals from every IT domain—network engineers, systems administrators, application analysts—sit around the conference table for hours attempting to prove that the problem isn't theirs. Whether you're an IT professional, or someone who directs teams of them, you know that any downed service immediately signals the beginning of a bad day.

The problem is that the idea of a "service that is down" is often so much more than a simple binary answer; on versus off, working versus not working. As you can see in Figure 1.1, IT services are made up of many components that must work in concert. Servers require the network for communication. Web servers get their information from application servers and databases. Data and workflow integrations from legacy systems such as mainframes must occur. These days, even data storage must be accessible over that same network.

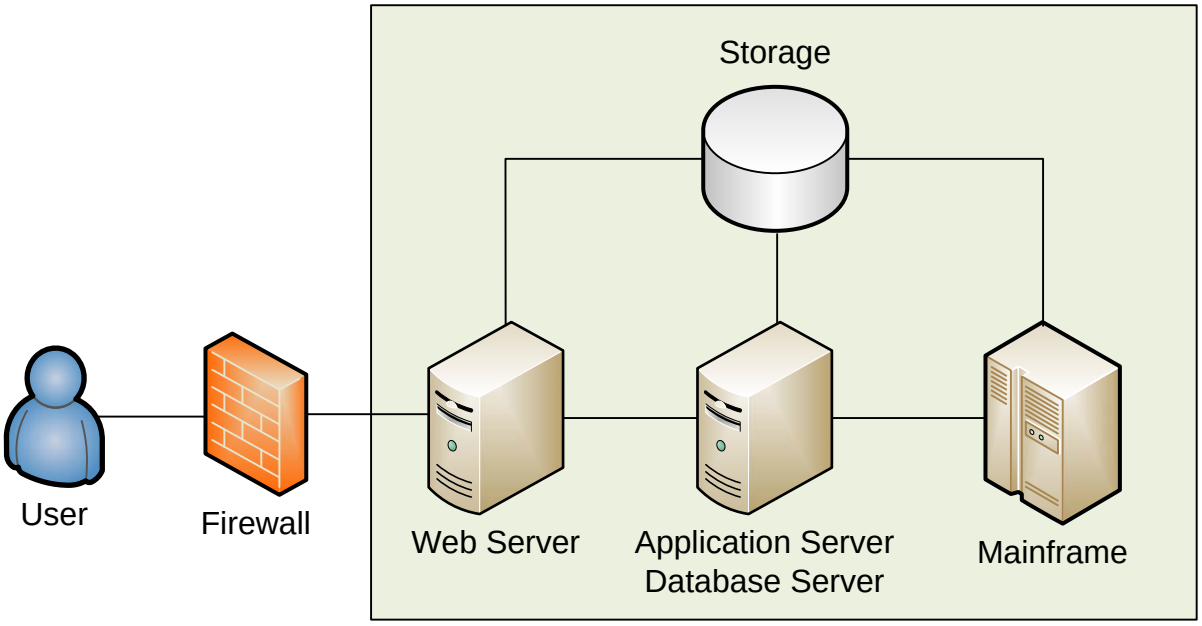

Figure 1.1: An IT service is comprised of numerous components that rely on each other for successful operation.

If any of those pieces experiences an unacceptable condition—an outage, a reduction in performance, an inappropriate or untimely response, and so on—the functionality of the entire service is affected. This can happen in any number of ways:

- The service or hardware hosting the service is non-functional
- A server or service that is relied on is non-functional
- One or more servers or services that make up the service are not performing at an acceptable level
- An individual component or function of the service is non-functional or is not performing at an acceptable level

All of these are situations that can and will impact the ability of your critical IT services to complete their stated mission. No matter whether the actual service itself is down or the cause is some component that feeds into the functionality of that service, the ultimate result to the end customer is a degradation in service. The ultimate result to your business is a loss of revenue, a loss of productivity, and the inability to fulfill the regular needs of business.

What Defines Functionality?

With all these components in play, actually defining what you consider "a fully functional service" grows much more complex. Consider some of the questions that you and your users are forced to ponder when non-nominal behaviors occur:

- If I can't access the service's Web site, is it functional?
- If I can access the service's Web site but am unable to login, is it functional?
- If I can access and login but am unable to complete a transaction, is it functional?
- If I complete a transaction but am unable to verify its completion, is it functional?
- If I experience an excessive delay in my completing a transaction, is it functional?
- If I can accomplish my tasks but my experience with the service is unsatisfactory, is it functional?

All of these are valid questions, because the mission of any IT service is to provide an expected level of value to its customers. That value comes from its ability to functionally complete a user request. It also comes from the ability to do so within a time frame that is acceptable to the user. The user must be able to interact with that service with a level of trust that their transactions have been successful and that they aren't wasting their time.

Unfortunately, the cultural history in many IT organizations hasn't always been so proactive in the identification and resolution of performance-based issues. It is the outwardly subjective nature of these questions why many IT services have had a tumultuous history with their customers. Over that short history, IT organizations have been notoriously simplistic in their view of service functionality. *Is the service on today?* If yes, then move on to the next problem.

For many years this binary view of the IT environment was sufficient for most businesses. As long as services were available, users could complete their goals. However, as businesses have over time grown more and more reliant on their IT services as a critical function of business, this immature "is it on?" approach to services under management can no longer be acceptable.

> **Cross-Reference**
>
> Chapter 2 will explore this history in greater detail and discuss how the maturity level of an IT organization bears heavily into how it goes about preparing for and solving problems.

What Is Application Performance Management?

Fixing IT's former "on versus off" approach to service management is therefore a critical step. As such, smart organizations are looking to accomplish this through a more comprehensive approach to defining their services, the quality of those services, and their ability to meet the needs of users. Application Performance Management (APM) is one systems management discipline that attempts to provide that perspective. Consider the following definition:

> APM is *an IT service discipline that encompasses the identification, prioritization, and resolution of performance and availability problems that affect business applications.*

Organizations that want to take advantage of APM must lay in place a workflow and technology infrastructure (see Figure 1.2) that enables the monitoring of hardware, software, business applications, and, most importantly, the end users' experience. These monitoring integrations must be exceptionally deep in the level of detail they elevate to the attention of an administrator. They must watch for and analyze behaviors across a wide swath of technology devices and applications, including networks, databases, servers, applications, mainframes, and even the users themselves as they interact with the system.

Figure 1.2 shows an example of how such a system might look. There, you can see how the major classes of an IT system—users, networks, servers, applications, and mainframes—are centered under the umbrella of a unified monitoring system. That system gathers data from each element into a centralized database. Also housed within that database is a logical model of the underlying system itself, which is used to power visualizations, suggest solutions to problems, and assist with the prioritization of responses.

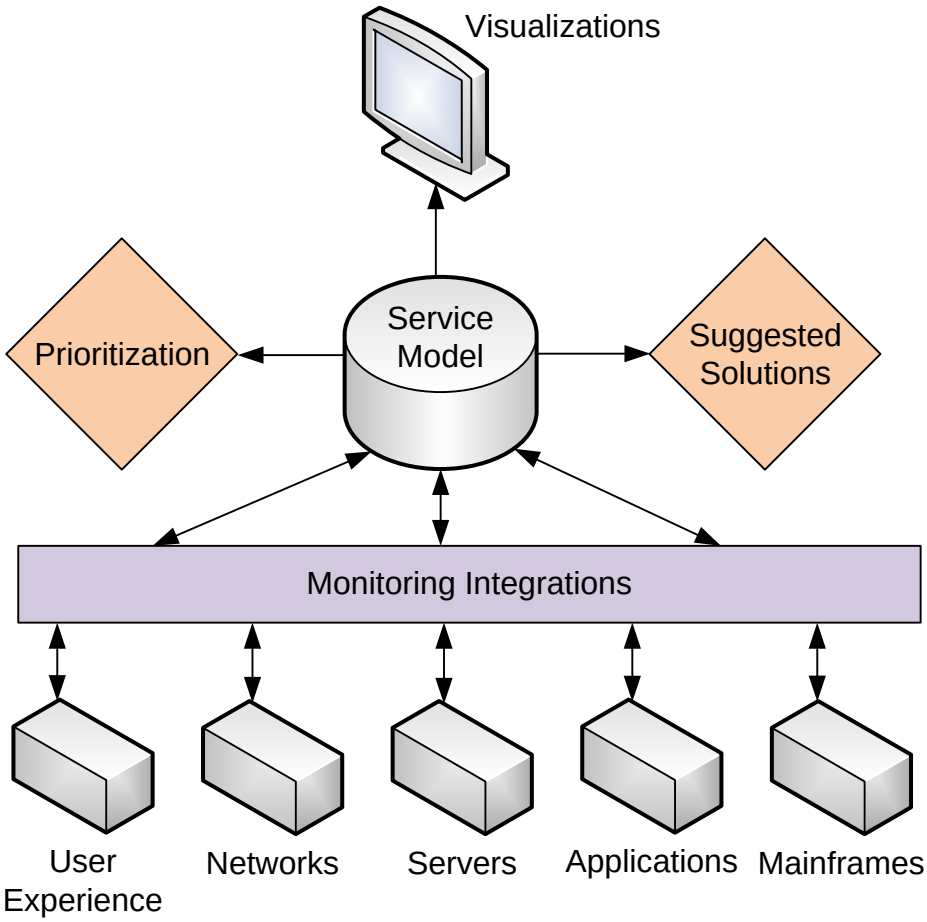

Figure 1.2: An APM solution leverages monitoring integrations and service model logic to drive visualizations, prioritize problems, and suggest solutions.

With its monitoring integrations spread across the network, such a system can then assist troubleshooting administrators with finding and resolving the problem's root cause. In situations in which multiple problems occur at once—not unheard of in IT environments—an APM system can assist in the prioritization of problems. In short, an effective APM system will drive administrators first to those problems that have the highest level of impact on users.

So, with this in mind, why are you here? Why read this guide?

The Goal of this Guide

The goal of this guide is to assist you with understanding the concepts and the promise that such a system can bring. A successfully implemented APM solution can bring that rich level of monitoring to your business-critical services. With monitoring integrations across every portion of your IT environment, APM can root out issues as or even before they become problems. An effective APM solution provides your administrators and your business with a situational awareness deep into the otherwise "black box" IT services you provide for your users.

These ten chapters are designed to help you understand what APM is and how it helps IT better align with the needs of its business. They will help you recognize the different types of APM integrations, including the all-important end users' perspective, and how the different integration types tie into that overall picture of service quality. They will give you an end-to-end understanding of APM in action, including the creation and use of interactive dashboards for users, administrators, and executives. You'll also come to understand how APM's information gathering brings data that can tie directly into business metrics through Business Service Management (BSM).

In the next ten chapters, you'll gain a detailed level of knowledge about the requirements and the power of APM, centered around the following topics:

- **Chapter 1: What Is APM?** This first chapter will document the problem with today's traditional monitoring solutions and explain why APM is an effective solution. It will discuss an introduction to APM and where it fits into the environment.
- **Chapter 2: How APM Aligns IT with the Business.** APM provides great value, but only when an organization's culture supports what APM has to offer. IT organizations must have a certain "maturity" if they are to get the highest levels of value from APM. Those organizations as they mature will, at the same time, find a greater alignment between their goals and the goals of their business.
- **Chapter 3: Understanding APM Monitoring.** As you've already learned, APM's monitoring integrations hook into vastly different areas within your IT infrastructure. This chapter will discuss those integrations, how they work, and how your existing monitoring infrastructure can augment an APM solution.
- **Chapter 4: Integrating APM into Your Infrastructure.** Following on Chapter 3's functional descriptions is a further explanation of APM's logical integrations into infrastructure components, network analytics, and application analytics. To build its holistic awareness of a business service, APM must peer into every facet of that service, from clients to mainframes and everything in between. This chapter discusses the nuts and bolts of how that will happen.
- **Chapter 5: Understanding the End User's Perspective.** You'll notice in Figure 1.2 that one critical component to be monitored is actually the end user. End User Experience (EUE) monitoring adds the users' perspective of the system, illuminating when users see problems that other areas of monitoring can't see.
- **Chapter 6: APM's Service-Centric Monitoring Approach.** With each of the necessary monitoring integrations in place, it is now possible to build a model of the service itself. This model creates a logical diagram of the service and its components, providing a map for monitors to display and update their information. The service model is also the structure that drives what kinds of data administrators will see within their assigned visualizations.

- **Chapter 7: Developing & Building APM Visualizations.** Within its visualizations is where an APM solution shows its power. These visualizations provide a heads-up display for notifying administrators when conditions are acceptable or unacceptable across the landscape of your IT infrastructure. This chapter will discuss ways to create useful dashboards for users, administrators, and executives.
- **Chapter 8: Seeing APM in Action.** With the information gained in Chapter 7, it is possible to see an APM solution fully in operations. This chapter will discuss how such a system works, how it enhances the processes for monitoring and troubleshooting, and how a fully-realized APM solution streamlines your steps to a problem's resolution.
- **Chapter 9: APM Enables Business Service Management.** BSM takes the technology focus of APM and relates its metrics to business goals. It adds a sense of financial logic to APM's availability, performance, and end user data, enabling business leaders to find and measure the value in IT projects and services. This chapter will discuss the linkages between APM and BSM and how the two work together for better IT alignment.
- **Chapter 10: The Shortcut Guide to APM.** Ten full chapters is a lot of reading material. Once you've read through the detail in these chapters, sharing that knowledge with others is key. That's the reason for Chapter 10. For those in your organization who need to know APM's concepts and its promise in a short form factor, this chapter quickly summarizes the key takeaways from this entire guide.

To crystallize your learning around real-world problems and solutions, each of the chapters in this guide (excepting this and the final one) will also include a chapter story. That story will discuss a situation that you've probably experienced either as an end-consumer, administrator, or director of an IT service. In that story, you'll meet a set of characters that will recognize the need for maturing their IT processes, decide to implement APM with its monitoring integrations, and ultimately build a successful system. Playing out over a series of chapters, you'll come to understand how APM can be hooked into an actual IT environment.

APM by Example

Now that you understand the path this guide will take in your understanding of APM, it is perhaps best to continue with a short example. This example is designed to tease your understanding of APM and prepare you for the chapters to come. Later chapters will continue with larger and more detailed examples like this one to assist with your learning.

Consider a business service that is similar to the one discussed in Figure 1.1. This system provides some level of service for customers on the Internet. It is made up of six major components: the network, network-attached storage (NAS), a firewall, and three servers that communicate with each other to fulfill the display, logic, and data storage needs of the system.

If this system is monitored through traditional device-centric management tools, certain problem situations are relatively easy to troubleshoot. If the Web server spontaneously powers down, a traditional management solution will quickly identify that the server no longer responds to ICMP ("ping") commands. Identifying the loss of an entire server due to a loss of power can be categorized as a "simple" problem, the resolution to which is easy to locate and apply. Power on the Web server, and you're back in business.

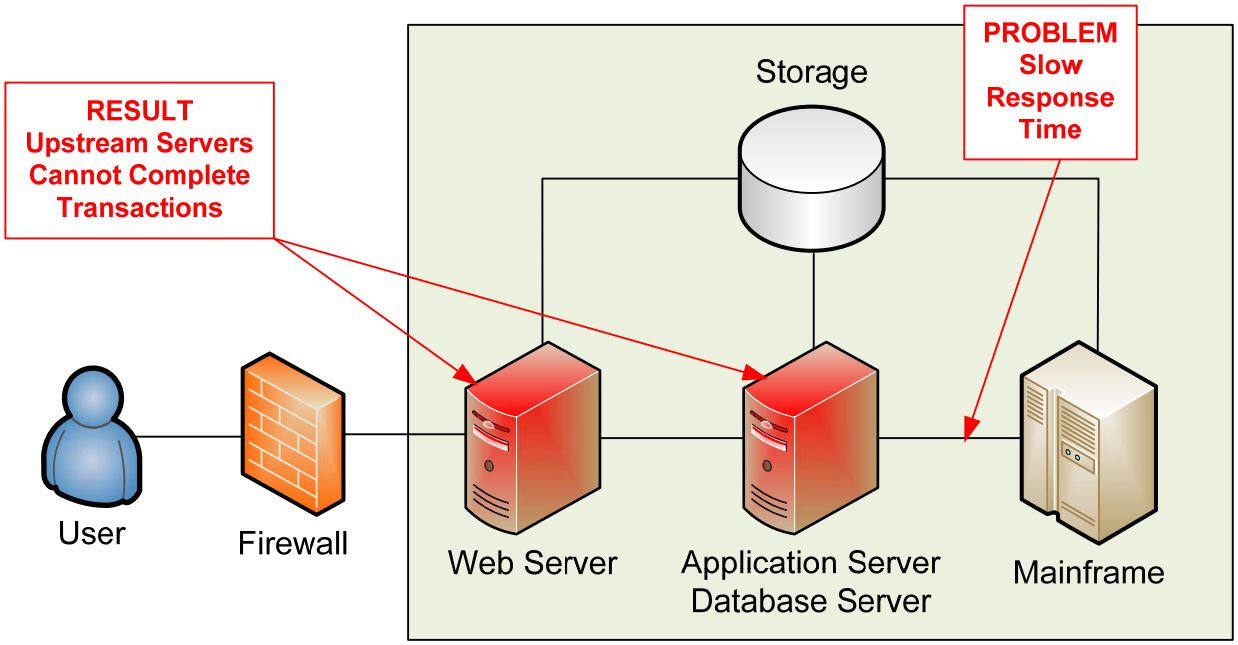

Figure 1.3: A problem deep within the mainframe can impact the operations of the entire system.

However, consider the situation in which the problem lies deeper within the application itself. In this example, the problem is not the loss of an entire server or device. Here, a much deeper problem exists. Rather than a simple server loss, the response time between the application server and the mainframe instead slows down. This occurs due to a problem within the mainframe. The decrease in performance between these two components eventually grows poor enough that it impacts the system's ability to complete transactions with the mainframe. As a result, the upstream reliant servers such as the application server, database server, and Web server can no longer fulfill their missions.

A problem like this is particularly difficult to troubleshoot because

- This situation doesn't involve the loss of a server or network device.
- It cannot be measured through onboard system counters that measure traditional metrics such as processor performance or RAM usage.
- It cannot be seen through network traffic analysis because the problem is involved with a reduction in traffic rather than bandwidth contention or latency.

Figure 1.3 shows a pictorial example of how this problem might manifest itself to the end user. In the picture, you can see how the information exiting the mainframe does not make its way to the application server in a timely fashion. Because of this slowdown in performance, transaction timeout values and thresholds are eventually exceeded. This causes the application server to no longer be able to serve the needs of the Web server, which itself can no longer serve the user. In the end, the user experiences a loss of service of the entire Web site, one that is difficult to trace back to the initial problem.

Why Measure Performance?

Why measure performance? Simply put, *because you must*. Measuring the performance of your servers gives you information about their internal workings. Measuring the performance of the network enlightens you to the levels of traffic going across the wire. Measuring database activity helps you understand the levels of resources that your applications require. Most importantly, measuring performance is a proxy for understanding business performance—the ultimate capacity of your business systems to service the needs of your customers.

Measuring application performance in the short run is a powerful troubleshooting tool, helping you identify where applications and infrastructures are not behaving properly. Long-term measurements are also useful in assisting continuous improvement teams with expanding existing systems. Looking at performance over the long-haul helps you understand when your existing infrastructure needs to scale.

Measuring Across Domains

The problem with most traditional monitoring solutions (see Figure 1.4) is that they're designed to work exclusively within a single problem domain. Network monitoring solutions give you information about network utilization but can provide no detail about processes on a server. Windows PerfMon counters can tell you the level of processor utilization and memory consumption but have no awareness of the underlying network conditions other than the data going in and out of its local network card.

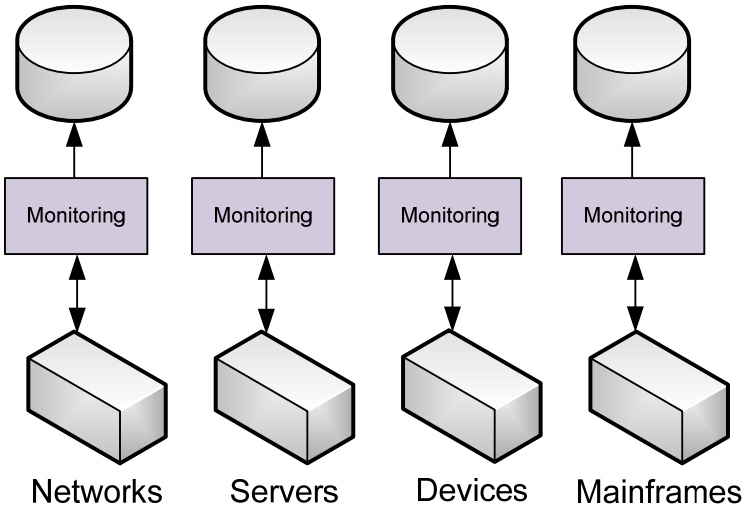

Figure 1.4: Stovepiped monitoring solutions don't provide a holistic view of the entire system.

Needed are solutions that integrate the monitoring information from each component. Such a system, like what was previously shown in Figure 1.2, will leverage the use of monitoring integrations across multiple domains into a single, centralized database for processing. In that database will be a model of the business service itself, creating the necessary logic that enables the system to understand and report on the data it is collecting.

Measuring Transactions

Yet this alone still isn't enough. Truly measuring performance is more than just enabling PerfMon counters on a Windows server or logging NetFlow statistics from a Cisco network device. Even comparing one device's set of information with another gives you only a limited perspective on the environment as a whole. Counters like these give you information about devices and interactions between devices. Nowhere in their data do they provide information about the applications that are installed atop those devices. Their aggregate data cannot illuminate the individual communications between, for example, two servers that comprise a service. To collect this data, it is additionally necessary to look at the individual transactions that occur between service components.

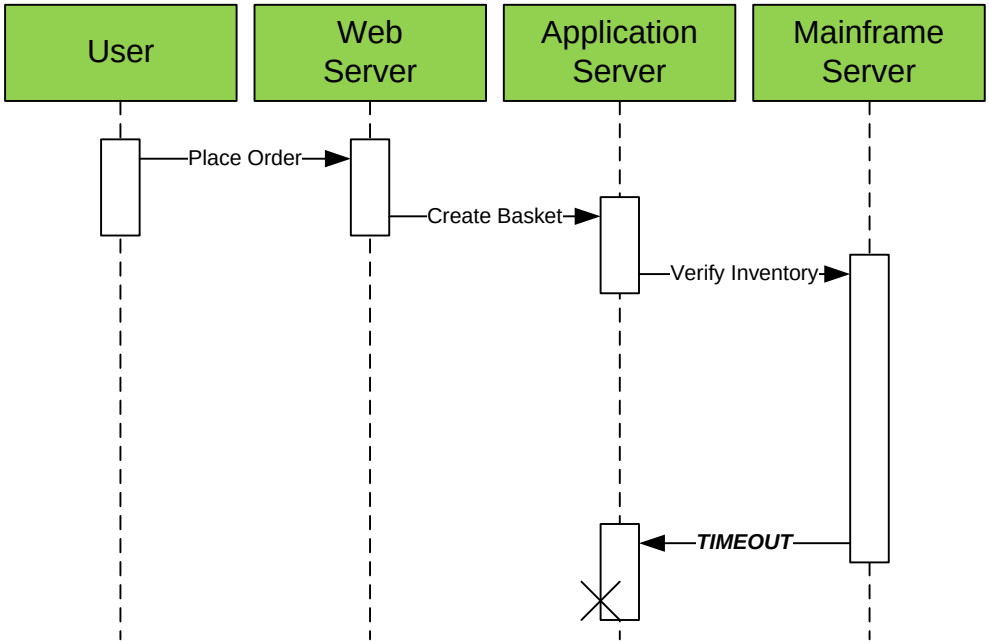

Figure 1.5: Transaction monitoring can watch the sequence of events between system components and alert when problems occur.

You've already seen an example of how transaction monitoring assists in the troubleshooting of a problem. In the previous section, a problem was identified where the communication between two servers, the application server and the mainframe, was impacted due to a problem with the mainframe. Although that specific problem on the mainframe might have been caught using traditional monitoring—perhaps because a daemon crashed or a socket stopped responding—traditional server monitoring solutions don't typically look at the communication that enters and exits a server. Thus, traditional monitoring solutions could never have correlated how the mainframe's problem impacts the other servers.

Figure 1.5 shows an example of how transaction monitoring might have recognized that a problem was occurring. The sequence diagram there shows the interactions between the different components that make up our example service. There, the user successfully attempts to place an order on the Web site. The Web site then attempts to create a shopping basket for the user. This action requires a verification of inventory levels prior to completion, an action that isn't completed in time. Consequently, the sequence ends with the application server timing out its request and an overall failure in the system.

Chapter 3 will go into more detail about this idea of transaction monitoring, followed by an in-depth discussion on its relation to the end users' perspective in Chapter 5. But for now, recognize that multiple and simultaneous mechanisms for monitoring business services are required if you are to obtain that desired level of situational awareness.

APM Optimizes the Application Life Cycle

Like projects, servers, and in fact the entire IT infrastructure, applications themselves experience a sort of life cycle all their own. With applications, however, the trendline associated with costs and benefits appears quite different than that for hardware and project resources. Applications, especially large-scale complex applications that are likely to be monitored through an APM solution, tend to have a life cycle that divides time between its cost period and its benefit period.

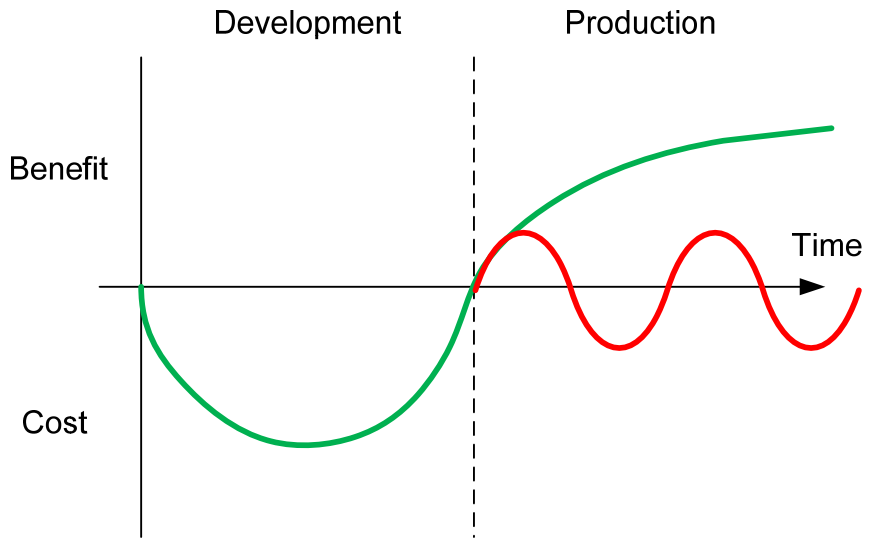

Figure 1.6: The application life cycle's impact to the business in both perfect and poor implementations.

Figure 1.6 shows an example of how the life cycle for an application can be recognized. Identifying the need, scoping the project, developing the application, designing its architecture, and eventually implementing the solution are all required cost elements that must occur to set the application in place. Once in production, "perfect" applications tend to require comparatively little marginal cost period over period.

For a perfect implementation, the impact to the business is displayed by Figure 1.6's green line. There, the application begins with zero benefit to the organization and all cost. Prior to its production deployment, no one within the organization is obviously making use of the application. At the same time, organizational resources are consumed for those aforementioned development activities. Once the application is brought into production, its benefits begin to increasingly outweigh the marginal costs required to keep it running. If that application is an internal tool for business employees, this benefit curve goes up as employees find value in it features. For external applications the benefit curve goes up as potential customers begin using the application and creating income for the business.

However, few applications experience that "perfect" curve between development and production, cost and benefit. The alternating red curve also seen in Figure 1.6 represents the fits and starts that occur with applications that are poorly brought into production. Perhaps the project wasn't scoped properly and more users are found to need the application than previously thought. Maybe the application creates an impact on the network that causes outages or performance issues with other services. Sometimes success itself becomes problematic. Newly-deployed applications can be so successful that the initial customer excitement over its release sends it down, crashing under the weight of its own exuberant users.

All of these represent areas in which issues with application delivery are caused through poor management of anticipated application performance. These issues occur both within the application as well as external to it. Too often performance is not considered as a primary requirement during application development, forcing performance management to occur late in the development life cycle. This omission of performance management early in application development adds cost to applications and extends their "cost" period. An added goal of APM is to provide the data foundation where performance impacts from the environment are understood at a macro scale. Here, monitoring integrations from all across the network environment are consolidated to provide the data necessary to create good designs with new applications right from the get-go.

Consider again the example from earlier in this chapter. In that example, some unknown condition on the mainframe was eventually found to be the root cause for a multiple-server failure. If fixing this problem requires a redevelopment of the core application, an outage like this represents a cost to the business. It means the trendline for the application switches from the "green" line to the "red" line, extending the time required to declare success with the application's deployment.

However, leveraging data gathered through an APM solution could have prevented such a problem from occurring in the first place. Perhaps the mainframe was already serving other customers and approaching the limit of its processing capabilities. Perhaps the piece of code built for the mainframe was not optimized in its processing requests, causing the mainframe to work particularly hard in processing every request. Any of these performance-related issues could have been potentially tracked down prior to the failure actually occurring.

So, How Does APM Work?

This entire guide is built to take you through the entire workflow of an APM implementation, from start to finish. But at a high level, an APM implementation starts with the deployment of a unified database into which information from each of the monitoring integrations will go. It serves as the location where metrics are fed based on behaviors that are monitored within the system. Individual monitoring integrations deposit their data into this location where it is processed and the useful pieces are stored. With so much data across so many devices and applications that can be collected, a major function of that database and its application logic is to identify what information is useful and what can be discarded.

How does the system identify what is useful and what isn't? Part of your implementation project will be to define just those parameters. This is done firstly through the creation of a logical representation of your services and infrastructure called the service model. As an extremely detailed picture, creating this logical representation can be one of the most challenging parts of an APM implementation.

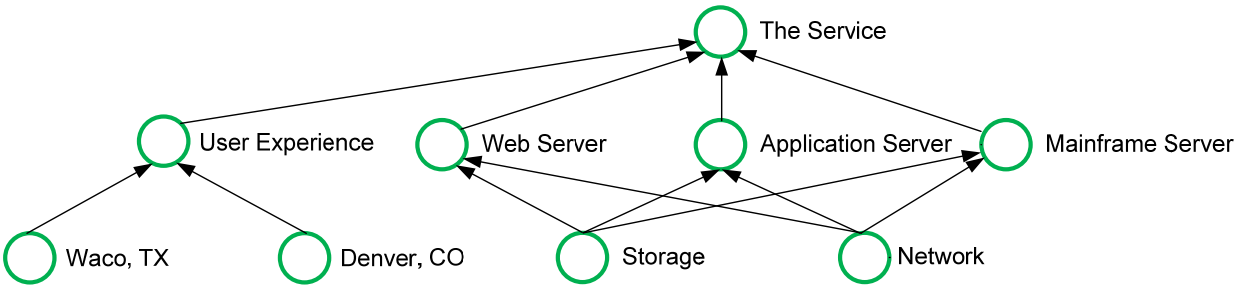

Figure 1.7: Mapping physical components into the service model's logical representation.

Installing an APM framework and its software results in what amounts to a blank canvas. On that canvas, you will sketch out your environment's architecture including all the elements that make up the systems to be managed. The completed service model becomes that logical representation of the components in your infrastructure. Figure 1.7 shows a simplistic example of how this might work. Here, each of the application's physical components and geographical locations has been mapped to a dependency diagram. The arrows in this dependency diagram show where elements within the model rely on other elements for their processing. At the top level, the service itself requires information or processing provided by the three servers in the environment. Each of those three servers requires the support of both network and storage components. Also shown is the user's experience divided between each of its multiple locations.

In this model, coloring is typically used to identify the health and status of each of its individual elements. You'll see that each of the dots representing an element is colored green. This easy-to-understand heads-up display provides a way to identify whether that component is functioning to desired levels. Obviously, when problems occur with a component, its green color will shift to yellow or red, denoting a caution or warning condition. Not shown in Figure 1.7 are the individual rules that are used in the background to identify the "greenness" or "redness" of the element. For example, underneath the network element may be custom-built rules that that identify bandwidth or latency conditions that are unacceptable or may indicate a pre-failure condition. When any of those rules are tripped by the monitor, its color changes to alert the problem condition.

A service model for any APM implementation is intended to be dynamic, organically changing over time as services or service components come and go. As the model grows in detail, it at the same time grows more powerful. Once defined, it graphically displays each of the dependencies that make up your application infrastructure. Should a dependent component experience a problem condition, the status of that component as well as those that depend on it can change. For example, the service's reliance on the mainframe means that any problem on the mainframe immediately rolls up to become a problem with the entire service itself. A problem with the network automatically impacts each of the servers' status, and ultimately the service as well. This roll-up and drill-down concept provides you with a mechanism to quickly trace to which components of the service are experiencing a problem, and drill down to the specific reasons underneath each element.

To obtain this data, multiple types of installed monitoring integrations are required. These integrations provide the hooks into various components that make up the service like code frameworks, databases, applications, and Web services. Installing and managing these monitors is a next big step in an APM implementation. Depending on the architecture of your applications and the components that make up the infrastructure, many monitors may be required to gather the right amount of data. Table 1.1 lists a few common monitors and integration points.

Web Sites	**HTTP / HTTPS**
	SMTP
	DNS
	OracleForms
	Siebel
	SAP
Framework / Analysis	**Message Queue / Microsoft MQ**
	XML/SOAP
	Tuxedo
	Citrix
	Windows Terminal Server
Applications & Databases	**J2EE**
	.NET
	Oracle
	SQL
	Sybase
	Informix
	DB2
Network Protocols	**NetBIOS**
	SMB
	DCOM
	RPC
	SSL

Table 1.1: A sample list of potential integration points for an APM solution.

Obviously, once installed, a major component of the day-to-day administration of an APM implementation is involved with the creation and tuning of the rules underneath each green dot. These rules identify the specific behaviors that your organization considers healthy versus those considered unhealthy for each component. They determine the state of each monitored system and when to alert that problem behaviors are occurring. Similar to the organic adjustments to the model itself, the creation and tuning of the model's underlying rules is also an ongoing activity.

The Role of Visualizations

The final component of an APM solution is the actual display of information itself. Although the service model provides one view into the health and status of system components and their linkages, it isn't necessarily useful for everyone. Needed is another layer of visualizations that leverage the structure and data of the service model but display it in a way that is more useful to its ultimate consumer.

This concept was first described in the book *The Definitive Guide to Business Service Management*. There the term *digestibility* is used to help the reader understand how different views make sense for different readers. This concept of digestible data relates to presenting the kind of data that is interesting, useable, and useful to its targeted class of individual.

Consider three types of individuals who may be interested in data that is generated by an APM solution. The first individual may fill the role of systems administrator. That individual may be interested in understanding large-scale conditions that occur across system components. They may have an interest in network conditions and the status of services that are in operations. When problems occur, they want to know specifically where that problem occurs so that they can drive towards a fix.

The second class of individuals who stand to gain through an APM solution are the developers of the system itself. A developer may be unconcerned with the day-to-day operations of a system once that system is in full production. Knowing which elements of the system are up versus down is not part of the job role of that system's developer. However, they do get involved when an administrator needs deep troubleshooting assistance with a problem, or when issues with the system require code updates or fixes. Application developers are likely to want to view more detailed information about the individual transactions between service components. They might want to see which transactions occurred versus which were unsuccessful. Information about the performance of page refreshes on an application's Web site is of much greater use to its developer, as that individual can find and fix the specific problem at the code level.

The third individual who might have interest is the end user themselves. End users of systems, especially those large-scale systems that are likely to be monitored with an APM solution, want to know when problems occur. This chapter started with a discussion of how users don't like seeing unhelpful messages like "This Web site is experiencing unexpectedly high volume. Please try again later." during an outage. An APM solution can be simultaneously used to notify end users when problems occur. It can be used to help them understand when performance conditions are lower than normal, and when users can expect to see a return to full operations.

Figure 1.8 shows a collage of potential visualizations that can be of interest to each of these classes of users. You'll see there the high-level stoplight charts that describe when system components are behaving at or below expectations. This view is handy for the administrator to know when system components experience problems. Also there is a detailed view of a set of individual transactions. This view provides the developer the necessary information they need to trace issues with performance or functionality. The third image shows an extremely high-level view of a global system, detailing areas where that system may be experiencing problems. This view provides the right level of detail for the end user, giving them the knowledge that problems exist.

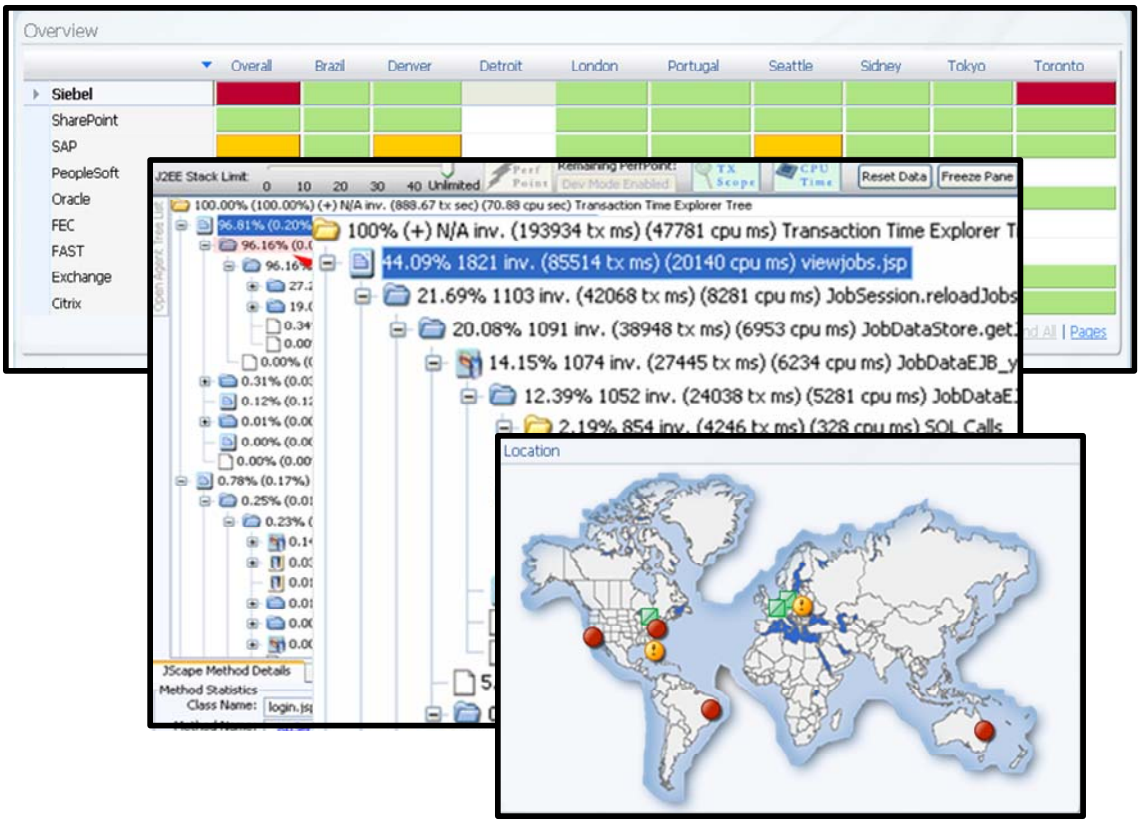

Figure 1.8: Three examples of visualizations that are tuned to the needs of their user class: administrator, developer, and end user.

What Benefits Does APM Provide?

Many of APM's direct benefits are easy to see just through the explanation in this first chapter. APM provides a level of awareness about IT services that is unmatched with today's technology. It hooks into virtually every component of your organization's application infrastructure, bringing back useful telemetry across each component into a single location for processing and later visualization. Breaking down these benefits a little further, consider a few of the extra benefits you can expect to see through a successful implementation.

Root Cause Identification

When an application or service experiences a problem today in your infrastructure, what is your first step in identifying that problem? Do you "circle the wagons," gather everyone in a room, and begin working through potential solutions? Do you reboot the servers and hope for the best? Or, do you leverage specific and actionable data that explains where on-system behaviors have changed?

All problems in an IT environment stem from some kind of change within the environment itself. This is the case because computers are deterministic. Some action or change must occur within the environment that drives the problem to occur. With the right monitoring integrations in place, it is possible to characterize the performance of your applications over time. It is then possible to use that characterization to track when an unacceptable behavior occurs, and immediately point the finger to where the root cause lies.

> **Cross-Reference**
>
> Chapter 6 will discuss this root cause analysis process in greater detail.

Characterization of Problems

Returning once again to the example problem in this chapter, characterizing the mainframe's problem was possible because its nominal behaviors were encoded into the APM solution's service model. Those nominal behaviors, such as acceptable processor utilization and acceptable delay in responding to inventory requests, provided a basis by which its later unacceptable behaviors could be alerted on.

When you've gone through the exercises necessary to characterize the acceptable behaviors of the elements in your IT environment, you provide that basis for alerting when unacceptable ones exist. Leveraging this ability with the dependency linking that makes up the service model provides a way to show how one component's behavior impacts others.

Prioritization of Problem Resolution

Each of the components that make up your application infrastructure is also likely to have a defined number of users. These users interact with the service to accomplish whatever tasks are automated by the system. Thus, when that system goes down, those users are prevented from accomplishing their tasks. When greater numbers of users are impacted by a particular component's failure, this increases the priority of fixing that component's problem.

A problem with this in many IT organizations today is simply not knowing how many users are affected by each business application or component. APM's structured approach to defining the overall architecture provides a way to easily roll up the number of affected users by component. This ultimately provides a way to prioritize which problems should be resolved first and which can be left for later resolution.

Situational Awareness

The data that passes along your network's wires is not something that can be directly looked at with the naked eye. Using traditional network sniffing tools to watch this data is also problematic due to the sheer quantity of data that flies by during any discernable period of time. Thus, better approaches that look at data from the network's perspective in combination with each server or application's perspective gives administrators a better situational awareness of what's going on in their networks. Combining this information with the rich monitoring support through any of APM's integrations means that the business can know the status of its applications at all times.

Better Planning

Lastly is the critical need for future planning. Too often, IT organizations go through planning exercises using a subjective approach, assigning augmentation dollars based on gut feelings or one-time problems rather than historical behaviors. Taking the long-term approach with APM data brings IT a mechanism for identifying where system elements need augmentation or wholesale upgrades. The data provided by APM integrations enables budgetary decisions to be made based on objective data.

Critical Applications Require Critical Monitoring

The goal of this chapter—and indeed this guide—is to help you understand the critical need for managing your applications' performance and behaviors. You've already learned the very basics of what is possible with APM solutions today. The rest of this guide will continue the discussion, with each chapter building on the information from the last.

However, before you can truly start your APM implementation, an important first step is in understanding your organization's level of process maturity. That level of maturity drives how you solve problems, how you react to situations, and what level of structure you have in place. You'll come to recognize that organizations who operate with a relatively low level of process maturity suffer under the weight of overwork, waste, and the lack of automation. Chapter 2 will discuss just those topics.

Download Additional eBooks from Realtime Nexus!

Realtime Nexus—The Digital Library provides world-class expert resources that IT professionals depend on to learn about the newest technologies. If you found this eBook to be informative, we encourage you to download more of our industry-leading technology eBooks and video guides at Realtime Nexus. Please visit http://nexus.realtimepublishers.com.