CHAPTER

MANAGEMENT OF TRADITIONAL APPLICATIONS

3

But that was yesterday and yesterday's gone…
Chad Stuart, "Yesterday's Gone"

A Day in the Life at Acme Manufacturing.

Meet Dave. He is a help desk technician working at the service desk at Acme Manufacturing. He has been doing this job for almost two years. From his perspective, it's just a job—definitely not something that he wants to do for the rest of his life, but for now, it pays the bills. His responsibilities are pretty limited; he spends most of his shift on the phone with users. He just finished a call that was fairly representative of what he deals with on a daily basis.

Dave: Service Desk, this is Dave. How can I help you?

Nancy: This is Nancy in billing. I'm not able to get into the accounts receivable application. In fact, nobody in the department is able to. What's happened to the application?

Dave: Are you able to get to the login screen?

Nancy: No, I can't get to anything! What's wrong?

Dave: Is anything showing on your monitor?

Nancy: Not much. I've got a black screen with a blinking cursor.

Dave thinks to himself, "Well, at least the monitor is turned on. It's surprising how often I get a call about a supposed network problem and it turns to be a monitor or PC that's not turned on."

Dave: Let me take a look.

The accounts receivable application is a relatively old (15 years), traditional application. It runs on a single server located in the data center, which happens to be in the same building as the billing department. Dave turns to the monitoring screen to see that the accounts receivable application appears to be running. Next, he looks at the network manager, which is not showing any signs of

Application Performance Management (APM) in the Digital Enterprise. http://dx.doi.org/10.1016/B978-0-12-804018-8.00003-6

trouble. Following standard procedure, he "bounces" *the line to the billing department. That is, he resets the circuit and the router in the department.*

Dave: Try it now. Can you get in?

Nancy: No. It's the same as it was before.

Dave: I'm going to have research this and open a trouble ticket. You'll be able to check on the status of the ticket at any time and I'll let you know when it's fixed.

Nancy: Okay, but can you make it a priority? This is the end of the month, you know.

Dave: Yes, of course. Goodbye.

At this point, Dave is honestly stumped. The tools that he has in front of him indicate that everything is working. Nothing seems out of the ordinary. Therefore, he escalates it to an application specialist. We'll return to Acme Manufacturing and the accounts receivable application problem at the end of this chapter.

Traditional **applications** are ones that were created 10 to 20 years ago, possibly even longer ago than that. Instead of calling them "traditional applications," most people in information technology (IT) refer to these applications simply as "really old," "ancient," or as a "dinosaur." These basic applications run on a mainframe environment or in a client/server environment. Architecturally, little has changed about them since they were created. They are old workhorses that have not been retired yet. They have not been "webified," componentized, containerized, or turned into a mobile app. Supporting them is not a highly sought-after programming assignment. Most developers are willing to do almost anything to avoid being assigned to maintaining one of these applications. However, they are a good place for us to start the discussion of **application management**, because in traditional applications things are (relatively) simple and easy to understand.

Managing an application consists of monitoring and controlling that application.

The most concise definition of **management** as it applies to the management of applications is that it consists of the monitoring and control of a managed object. The managed object can be anything. It may be a router, a server, or anything else that can be monitored and controlled. In this case, an application is the object that is being managed.

Saying that management consists of a monitor function and a control function is useful as a quick pneumonic, but this hides the complexity required in order to effectively manage applications (or any other object). The monitoring and control of an application is not achieved in a vacuum. They require the use of people, processes, tools, and technologies to achieve those functions. However, that is not enough. Before an application can be managed, it is necessary to know that it exists, where it resides, when it is running, and where the code can be found when it is not running.

LOOKING BACK

In the early days of IT, knowing which application was running was the purview of the **master console operator (MCO)**. Most of the knowledge about applications was contained in run books, shift turnover notes, handwritten notes passed from one MCO to the next, and general tribal knowledge. The environment was relatively simple and stable (although at that time it did not seem to be either simple or stable). Jobs were run in batch mode, and "everyone" knew which hard drive held a particular application. Applications could be configured and executed through a set of instructions written in

job control language (JCL). The MCO monitored the status of each application, and there were really only two results of running an application: the application could complete successfully or it could end abnormally (ABEND). The universe of applications that might be run in a medium-to-large data center typically numbered in the low hundreds and the dependencies between applications were almost always linear. For example, the order entry application ran before the accounts receivable application and that always ran before the general ledger application. Components of an application ran on one system. That is, various components of a single application were not dispersed and did not run on multiple computer systems. In fact, there may have been only one system in the entire data center.

Today, the IT environment is an infinitely more complex atmosphere. A large company will have thousands of unique applications—tens of thousands, in some cases. Applications are no longer confined to running on a single mainframe computer in a data center. Servers are scattered far and wide across the enterprise, and some of them are not even properly accounted for (i.e., are not recorded as an asset on the company's books). Some may have been purchased surreptitiously, perhaps even recorded as "HVAC repairs" or "warehouse shelving units." While such slight of hand may be helpful in concealing a server from the powers that be for a while, it is inappropriate and, in some jurisdictions (including the United States), illegal. This same practice applies to applications. That is, the purchase of **commercial-off-the-shelf software (COTS)** is concealed as a different type of expense. Staff responsible for developing applications internally may be hidden by classifying them as regular **line of business (LOB)** staff. Regardless of the ethics or legality of such practices, they occur all too frequently. The result is that it is unlikely that IT is aware of all of the systems that are running in an enterprise and even less aware of all of the applications.

Once systems moved out of the tightly controlled environment of the data center, the genie escaped from the bottle, never to return. Since that moment, it has been impossible for operations staff—whether in the IT department or in a business unit—to know with certainty what applications exist within the organization, where applications have been installed, where they will run (since not all of the pieces of an application will run on the same system and one piece may not even run on the same system every time), when they were installed, or when they were modified. Operations personnel may or may not know when a particular application is supposed to run, what dependencies it may have for other applications or resources, or even the owner of the application. It is something of a Wild West environment.

Add applications that are being run on desktop/laptop systems to this chaotic world. Many people tend to trivialize the significance or difficulty of managing the applications running on those systems, and some even question the need to manage them at all.

First, let's make a distinction between those desktop/laptop systems that are running a minimum set of productivity tools (e.g., Microsoft Office and a web browser) and those that are running a more complex set of applications. In truth, those plain vanilla desktop/laptop systems are rare in the business world. Even your elderly Aunt Martha will probably have a few extra applications that she uses on a regular basis (e.g., an eReader, an application to archive her photos, perhaps an application for playing solitaire, etc.). When considering the relative ease of using laptop/desktop systems and of installing or even upgrading applications on them, it is important to remember the sheer volume of objects to be managed and that they are often not connected for extended periods of time. Adding to the difficulty is that, universally, users seem to be incapable of resisting the temptation to make changes to their system, alter the configuration of the applications, and or installing their favorite applications. In short, what may start out as a **personal computer** with a standard set of applications configured according to a corporate standard very quickly is modified and customized according to the whims of the user.

THE MANAGEMENT IMPERATIVE

Any system, and particularly the applications that run on it, for which technical support is provided must be manageable. In conjunction with work on standards for **Open Systems Interconnect (OSI)** and **Common Management Information Protocol (CMIP)**, the **International Standards Organization (ISO)** defined management as consisting of the following functions: **fault**, **configuration**, **accounting**, **performance**, and **security (FCAPS)** (Fig. 3.1).

Any system, and particularly the applications that run on it, for which technical support is provided must be manageable.

The first step in managing an application on a desktop/laptop system is to configure the system. The initial configuration may define the user, define the security policy (e.g., is a password required to access the total system or an individual application?), and set up the email system. In a corporate setting, it will be necessary to capture some information about this asset. This will be used for inventory and depreciation purposes (accounting) and as a reference for technical support personnel (fault and performance). We will examine application management of a desktop/laptop system in greater depth

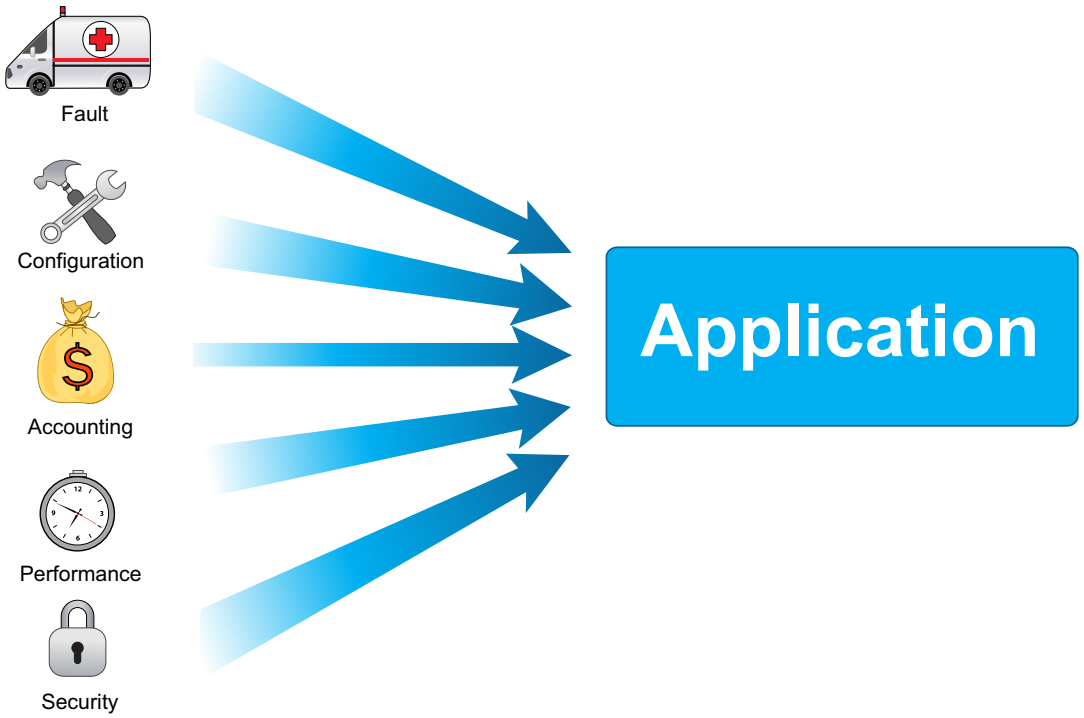

FIGURE 3.1

ISO management functions.

FIGURE 3.2

Mobile devices: the "elephant in the room."

later in this book. The key point to take away at this stage is that application management is required in these environments and it has special challenges. The era of **bring your own device (BYOD)** has further compounded the challenges of application management. We will look at those issues in Chapter 6, "Management of **Mobile Applications**."

We talked about the management of applications on **mainframe computers**, servers, and desktop/laptop systems. Next, we need to talk about the elephant in the room (Fig. 3.2): **mobile devices** (e.g., smartphones and tablets). The proliferation of mobile devices represents a new environment with management challenges that are materially greater than those seen in the other environments. All of the issues associated with applications on laptop computers exist with mobile devices in a plethora of unique **operating systems**. We will look at the management of applications on mobile devices in more depth in Chapter 6.

Taken together, application management across the various platforms is a dynamic and exciting environment. Business units are getting more benefits more quickly from the computing resources that they rely on. They have found that they can connect more tightly with their customers—increasing customer satisfaction and "stickiness" (i.e., customer loyalty).

With multiple platforms, however, comes the clear problem of how to manage them all. What we need to do is to learn to apply the principles of application management across all of these environments to maximize the performance and availability of those applications while, at the same, maintaining an acceptable level of security.

RESPONSIBILITIES

We can all agree that technology has reached a point where application management is very difficult, so how can we turn it into a manageable task? The answer is simple: break it down into smaller pieces and apply automation (Fig. 3.3).

Another way of looking at application management is that it requires two things: to know (monitor) and to do (control). Who needs to know about an application? First, there are the frontline staff who are

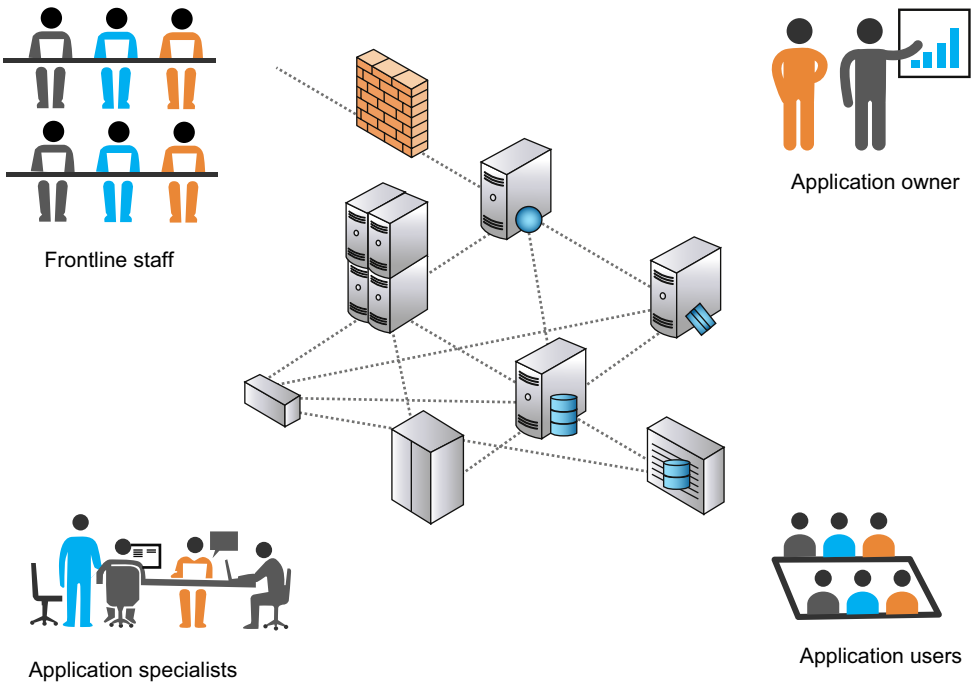

FIGURE 3.3

The players.

responsible for keeping things up and running smoothly. They have a variety of titles: system administrator, help desk technician, service desk analyst, etc. The exact titles will vary, but these are the people who actively monitor environments for problems; who receive alerts from management software and who are charged with resolving those problems when they occur. They are the firefighters.

There is another group of people who address application management problems reactively (more difficult problems are referred to them for resolution) and proactively (taking steps to prevent problems from occurring). There is not a lot of consistency in titles for people in the second group. For this discussion, we will simply refer to them as application specialists. They also install and configure applications and have the seemingly endless challenge of making sure all copies of an application are the current version. Maintenance agreements for third-party software must be kept current and updates applied when appropriate. Just maintaining an inventory of software installed and software licensed throughout the enterprise is a daunting task. Another challenge is making sure that the business has current licenses for every copy of third-party software that is in use. Finally, there are the application developers—the people who develop and maintain the applications (please note: this list is not intended to be an exhaustive list of job titles and responsibilities. Titles can vary significantly from one organization to another).

There are a couple more players that deserve to be mentioned. They are the application users and the application owner. The users are the people who actually use an application to perform some

business function. They may be employees, customers, contractors, etc. The application owner is the business executive whose staff or customers are the primary users of the application. In some organizations, the application owner funds the development and maintenance (or purchase) of the application and may have some degree of limited visibility into the operational status of the application. However, neither the owner nor the user has any responsibility or authority over the management of the application.

KNOWING

Let's take a look at what each of these groups needs to know.

FRONTLINE STAFF

This group needs to know what is happening at any given point in time. They need to know the health of an application, essentially, its availability and performance. The data can come from a variety of tools, plus reports from users of the applications. The following are examples of what they need to know:

Installed applications
Application dependencies (other applications, resources, etc.)
Applications that are currently running
Applications that are supposed to be running but are not (e.g., failed, waiting on other jobs or resources, etc.)
Indicators of the **application performance** level
Application availability (running does not mean an application is available to users)
Configuration settings
License information (including status of maintenance agreements)
Ecosystem information (network, systems, databases, etc.)

As discussed in Chapter 12, a user does not see each of the unique pieces that must work together in order for that user to be able to access the application's functionality. If I am a user, I look at availability as a binary matter. I can either use the application to do my job, or I cannot. If I go to an ATM to conduct a transaction, perhaps to withdraw cash, it either works or it does not. I am totally unaware of the multiple applications that are involved in serving my request. As a user, if I cannot access the application, I do not know nor do I care what the cause is; I just want it fixed immediately. It is that simple. It is similar when thinking about performance. The application responds quickly, slowly, or somewhere in between, but I am solely focused on my use of the application. If the application is not available or its performance is degraded, I do not care if the cause is a network problem, a server issue, or a change to the configuration of the application itself. I just want to be able to get some cash so I can go to dinner with some friends on a Friday night (Fig. 3.4).

There are a myriad of things that can impact the performance or availability of the application. The management of those things that are outside of the application (e.g., network, systems, data, etc.) is beyond the scope of this book.

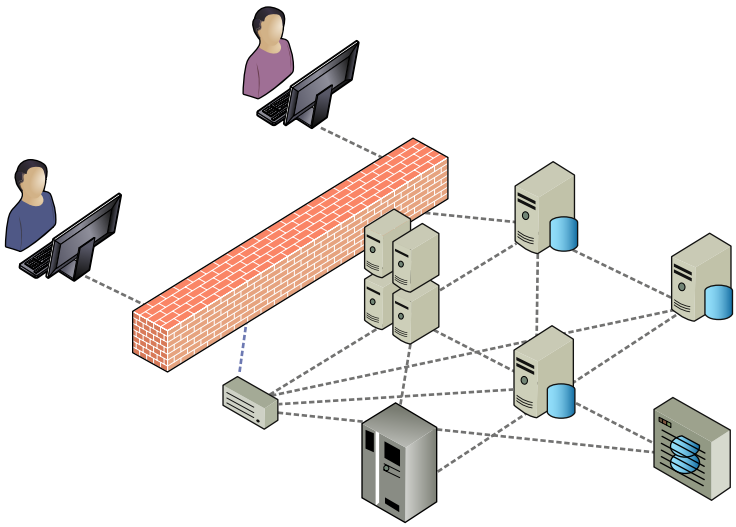

FIGURE 3.4

User perspective.

APPLICATION SPECIALISTS

Their information requirements are similar to those of the frontline staff, but they are much more concerned with information about performance, configuration, and application dependencies. While the needs of frontline staff are primarily for real-time information, application specialists tend to work from a historical perspective (except when they are installing and configuring applications). The following are key items required by application specialists:

Installed applications
Application dependencies (other applications, resources, etc.)
Indicators of the application performance level
Application availability (running does not mean an application is available to users)
Configuration settings
License information (including status of maintenance agreements)
Ecosystem information (network, systems, databases, etc.)

APPLICATION DEVELOPERS

Their need for application management capabilities parallels those of their colleagues in operations, the frontline staff and application specialists. This is because their responsibilities are not just limited to writing code. They must also test the applications that they write (or modify) before moving them into the production, meaning that they need to be able to install and configure the applications in a test environment. They also need to be able to monitor the application while it is running to see how it performs, which lets them see what happens to it under a workload similar to what will be encountered

in production. In some situations, depending on company policies, application developers may be asked to help resolve a problem with an application that is in production. In that situation, they may need to access some of the same management tools in the production environment used by frontline staff and application specialists. It is also the responsibility of application developers to create applications that are manageable. We will look at that set of challenges in Chapter 9, "**DevOps** and **Continuous Delivery**."

It is in the realm of *knowing* that automation is most important. The computing resources in even a small enterprise are capable of generating a flood of data far beyond the ability of any human to absorb and analyze. Management tools are the only way to address this problem, but they are both a blessing and a curse. The strength of management tools lies in their ability to collect huge amounts of data, and then analyze, summarize, and prioritize it before presenting it as information to be acted on by the humans charged with managing the applications. This is the positive side of management tools. The negative side is that the tools are capable of collecting so much data (even when summarized) that they can flood their human counterparts. It requires both art and science to select the right management tools and to configure them optimally to ensure that the people managing the application(s) receive all of the information that they need without being buried with a flood of data.

CONTROLLING

Each of the players in the application management process needs to be able to exert nearly absolute control over the application(s) that he or she is monitoring. They have the authority to start (or restart) applications, cancel an application, modify configurations, install new versions, apply patches to the application, etc. However, only application developers have the authority to modify the application's source code. Changes to source code will normally be made separate development environment and then tested in a separate "test" environment before being moved into the production environment. When there is a crisis, though, some of the best practices may be bypassed in the rush to resolve a problem as quickly as possible. While the urgency is understandable, bypassing standard processes carries the risk of introducing new problems or creating gaps in the application's security.

In most cases in an operational environment, frontline staff and application specialists will rely upon management tools as the medium through which they exercise control of applications. While there are occasional exceptions, the use of management tools is the most efficient way to exercise control over an application and, in general, is the only practical way to do so. As an example, think of software distribution. In a single enterprise there can be thousands of systems running pieces of an application that are often scattered across many countries. Historically, installing a new application or new version of an application required someone physically going to that device and installing the software. However, in a distributed organization with thousands of systems in dozens to hundreds of locations, the cost of sending a person to each of the enterprise's locations is prohibitive. The logistics are also impractical, particularly in those cases when it is necessary to have all of the instances of the application cut over to a new version at the same time. In response to those problems, some organizations tried the "honor system" of software distribution. That is, the new software would be shipped to someone at each location. That person was given instructions about when and how to install the software.

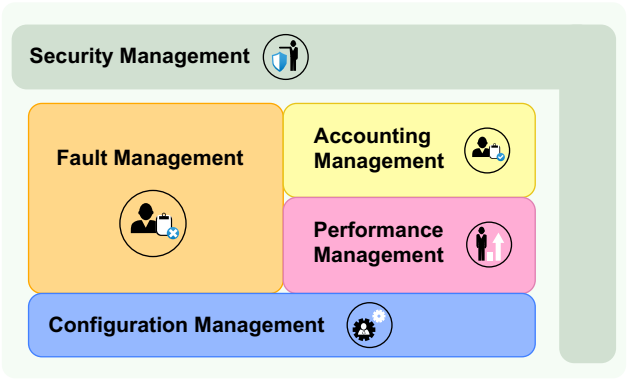

FIGURE 3.5

FCAPS model.

In theory, the "honor system" for software distribution sounded like a great idea. However, the theory ignored some simple problems, such as some people were traveling and did not receive the software before the deadline for the installation. Others felt that they had more important things to do than installing the software. In short, for a multitude of reasons (or excuses), the "honor system" for software distribution was a failure. The solution to the problem was to automate the process. Today, software is pushed to distributed systems by a software distribution system. The tool for software distribution may be a COTS product or a tool that was developed in-house.

FUNCTIONAL PERSPECTIVE

Earlier in this chapter we discussed the ISO model for management. Let's take a quick look at how those functions are realized in a traditional application management environment (Fig. 3.5).

FAULT

A management tool monitors each application for any indication of a problem with the application and frequently polls the application for information about the application's status. When a problem is detected, frontline staff will be notified. Ideally, the management tool detects the problem and notifies them early enough that failure of the application can be averted. However, too often that does not happen. Perhaps there were no early warning signs of the impending failure of an application, perhaps staff was not paying attention, or for a myriad of other reasons an application can fail and sometimes have a cascade effect on other applications, causing them also to fail.

CONFIGURATION

This is commonly done with administrator authority for the application via an interface created by the application developers for this purpose. Unfortunately, creating that interface is often done at

the last minute and done poorly. Sometimes, but not always, this results in an interface that is arcane and difficult to use. Fortunately, more experienced developers recognize the importance of that human interaction and deliver an interface for application configuration that is more user-friendly.

It is in the configuration process that the application's operating parameters are set. Those parameters include such things as:

Users and their respective authority levels, a.k.a., what can each person do?
Data sources/repositories and the location(s) of them
Dependencies to other applications
Port mapping
URL specification
Network configurations

Some applications need to have multiple network cards installed on the server for additional bandwidth or parallel processing. It is also important to specify such items as **quality of service (QoS)** priorities so network settings can reflect how the application actually works. Anything involving video or even audio would need higher priority, while bulk data exchange or transfers can often proceed at a slower rate. Also, if a **virtual private network (VPN)** connection is required, that needs to be defined.

Getting the configurations right is critically important. Mistakes in configurations are a common source of application-related problems; for example, where mismatches occur between "as designed" and "as provisioned."

> Note: It is important to recognize that the term "configuration" in the context of this book refers specifically to the operating parameters of an application. In the world of application development, "software configuration management" refers to the processes put in place to manage the introduction of changes to an application.

ACCOUNTING

This aspect can be divided into two primary functions: **asset management** and resource utilization. Asset management requires awareness that the application is present in the environment. That should be easier with COTS applications, or at least that is the theory. In practice, the location of each application can be very loose. Operations personnel will move applications from one system to another for any myriad of reasons, such as to balance the workload between systems, as a workaround to a problem, etc. They will sometimes install a copy of a COTS application on another system as a backup in case something happens to take down the "live" copy. In that event, they can quickly start the backup copy, which is a sound practice. However, it often occurs that the software license agreement for that application does not allow for the backup copy to be created, and by making a backup copy the staff has created a legal problem for the enterprise. Before installing a backup copy of a COTS application, it is important to make sure that license for that software allows for a backup copy.

There are two options for keeping track of applications. The first is to rely on the "tribal knowledge" of operations personnel. That is, rely on the IT organization to keep track of what it has and where each application is located. While prone to errors and omissions, this approach is widely used because it is cheap and because it has inertia on its side, i.e., the mindset of "we've always done it this way." The other option is to regularly use a management tool to discover the applications. This approach has its own challenges—especially if operations staff have attempted to hide an application. It is not sufficient to know where an application exists or even what dependencies it has. The application's attributes need to also be captured and documented. This can happen as part of the configuration management process or it can happen separately, but it must happen. More mature IT organizations store that information in a **configuration management database (CMDB)**. An overview of CMDB is provided in Appendix B.

Information about resource utilization is obviously important to operations personnel (including frontline staff and application specialists) in their daily activities of monitoring the application for problems, in performance, and tuning. Resource utilization data are also important for capacity planning and for anticipating when additional hardware will need to be added.

PERFORMANCE

Fault management and **performance management** are the two domains that account for most of the daily activities of frontline staff and application specialists. While fault management looks at an application to determine whether it is "broken" (i.e., it has failed or has another serious problem), performance management is concerned with keeping an application running "well." That is, performance management seeks to ensure that an application is operating within the parameters specified in the relevant **service-level agreement (SLA)** and **operational-level agreement (OLA)**. This is largely a question of the speed at which the application does its work. An absolute requirement for **application performance management (APM)** is a set of tools to collect and analyze data about the application. Some of the collected data will be archived to allow a historical perspective to be taken when analyzing problems. SLAs are discussed in Appendix A, "**Service-Level Management**."

SECURITY

The priority for application security is ensuring that only authorized people or other applications are allowed to make use of the application's functionality. That is, only those with permission can use the application to retrieve information or instruct it to perform an activity, such as generate a refund to a client, order more supplies, etc. Preventing intrusions into a system, while absolutely essential as part of an overall security strategy, is not part of security for the application. The role of application security is to ensure that if a hacker gains access to a system, the intruder is not able to use the application. Another piece of the application security puzzle is ensuring that only authorized changes are made to the application or to its configuration (change control). Without that protection, an intruder or a disgruntled (or inept) employee can simply change configuration settings in ways that compromise the integrity of the application. Securing applications is discussed in Chapter 8, "Application Security."

A Day in The Life (continued)…

When we left the accounts receivable application problem, it had just escalated. When the escalated ticket is referred, it gets routed (randomly) to an application specialist. In this case, it landed on the desk of Liz Winters. She has been working at Acme 12 years, eight of them in the IT department. She likes working at Acme, since she has a lot of friends there. She also enjoys the variety of challenges that her job presents.

She has a lot more experience, training, and resources than Dave, so she begins by making sure that the network is really up. She tries to ping the router in the department, which is successful.

"OK," she thinks. "It's not the network."

She can see that the accounts receivable application is up. However, when she looks at a system monitor, she can see that the application shows zero activity. She wonders, "What's going on here?" As she drills down further, she is able to see that there is no I/O activity for the application.

"Hmmm, that's odd."

Hoping for a quick fix and acting on a hunch, she decides to cancel and restart the application. A few minutes later, she checks back and is surprised to see that there is still no I/O activity for the application. Not wanting to waste time chasing phantoms, she calls Nancy in the billing department and asks her to try to log in. Unfortunately, Nancy still cannot.

Now Liz is puzzled.

"The network is up. The system is up. The application is up, but it isn't doing anything. Okay, at least at a theoretical level. I'll buy the idea that without anyone using the system there wouldn't be any noticeable activity, but why can't the users get into the app?"

She talks with a couple of coworkers. One of them suggests checking the configuration for the application and if that appears normal, then it will be necessary to call a developer for help, which she hates, as she feels that some of them are really condescending toward the application specialists. Before she does this, she pulls up the documentation from the CMDB system so she can see what the configuration settings should be. The configuration for the Accounts Receivable application is split into several files.

This is going to be a slow, methodical process. She is going to have to check every value for every setting in each of the files. There cannot be any shortcuts here. The only thing that can shorten the process is being lucky and finding the problem sooner rather than later. Before she can start, her manager calls and asks for an update. Accounting has escalated the problem at their end and now the operations director wants to know what is going on and how long it is going to take to fix it, etc. She

grumbles to herself, "I'd be closer to solving this thing if they'd just let me do my job instead of sitting here explaining everything."

Explanation finished, she starts through the first config file. Based upon what she knows, she decides to start where there is a higher probability of finding the cause of the problem: the sysconfig file. As she starts checking the settings, she notices that the setting for the address of the AR database is blank.

"What the…?" *she thinks.* "What moron did that?"

She quickly plugs in the correct value; kills the accounts receivable application, and restarts it. While waiting for it to come back up, she reflects on how the IT department at Acme Manufacturing could be so progressive in some areas and foolishly stingy in others. Because there is not a tool in place to track and manage changes to the configuration settings, there is no way that they would ever know who deleted the setting for the location of the AR database, unless the person who did it was foolish enough to confess.

The accounts receivable application is back up. Liz checks her monitor. The accounts receivable application is now showing a low level of I/O activity. Although the application had appeared to be active before, it had never completed its startup routine, and that is why no one could log in to the application. She picks up the phone and calls Nancy in the billing department to give her the good news. Total downtime charged against the SLA: 1 h 12 min.

As she proceeds to close the trouble ticket, Liz makes a promise to herself that at her manager's next staff meeting, she should bring up the problems caused by the lack of a tool to track and control changes. It is then that a troubling thought crosses her mind: what if the problem had not been caused by a mistake; what if had been done intentionally? There is no way to know – and that will be her main message at the next staff meeting.

This is a simple example of how application management can be critical to IT being able to meet the needs of the business. We will meet the folks at Acme Manufacturing again, later in the book.

SUMMARY

Managing traditional applications is not rocket science. In fact, it is relatively simple when compared to more modern application architectures. However, it still requires a structured approach to the process, well-trained staff, and the appropriate tools. Without those tools, application management would be reduced to waiting for an application to fail or for its performance to degrade and then trying to guess at the cause as well as a solution. That is simply not tenable.

KEY TAKEAWAYS

- Application management consists of monitoring and controlling applications
- ISO model for management
  - Fault
  - Configuration
  - Accounting
  - Performance
  - Security

- Even the management of traditional applications requires tools
  - **Real-time monitoring** of the applications (fault, performance, and security)
  - Data collection, analysis, and storage
  - Application Discovery and Dependency Mapping
  - Change control
  - Service desk management tools