CHAPTER

THE EVOLUTION OF APPLICATION MANAGEMENT

2

Living systems are never in equilibrium. They are inherently unstable. They may seem stable, but they're not. Everything is moving and changing. In a sense, everything is on the edge of collapse.
Michael Crichton, Jurassic Park.

One general law, leading to the advancement of all organic beings, namely, multiply, vary, let the strongest live and the weakest die.
Charles Darwin, On the Origin of Species.

Authors' note: *Some readers may be tempted to skip this chapter, feeling that the evolution of application management is a look into the past with little relevance to the challenges faced with application management today. However, this chapter is important because just as the roots shape a tree, the technical decisions of the past shape the future directions of technology. While it is not necessary to memorize every line of this chapter, we encourage you to spend a few minutes with this section. It will pay dividends in the subsequent chapters.*

In the above quotes, Crichton and Darwin understate the scope of change. Change is not limited to living systems. The entire universe is in a constant state of flux. Stars are being born while others are dying or exploding. If you stop and reflect, you can surely think of at least a few things that seem unchangeable. However, that appearance is the result not thinking in the long term—the *really* long term. For example, the movement of tectonic plates continuously reshapes the surface of the earth, but it happens not even for generations, but rather over millions or billions of years.

The biases of Victorian England influenced Charles Darwin. He presumed that evolution, through survival of the fittest, would inevitably lead a better species—to a better state. Unfortunately, that is not true for plants and animals, and it is definitely not true for computer systems.

HISTORICAL PERSPECTIVE

To understand the evolution of **application management**, it is necessary to first look at the evolution of computers and of the **applications** that ran on them. Some people will argue that the Jacquard loom deserves recognition as the first "computer." However, that, at best, could be considered the forerunner of numerically controlled manufacturing equipment. The looms did earn distinction for their use of a series of punched cards to control complex operations by the looms. While those series of cards bore a resemblance to paper tape that was used two centuries later, the looms did not process data (Fig. 2.1).

Application Performance Management (APM) in the Digital Enterprise. http://dx.doi.org/10.1016/B978-0-12-804018-8.00002-4

FIGURE 2.1

Jacquard loom.

Of course, some argue that the first programmable computer was either the Difference Engine or the Analytical Engine designed by Charles Babbage in the mid-19th century. Both were the result of the sheer genius of Babbage. Unfortunately, only small parts of the machines designed by Babbage were built during his lifetime. Despite that, they represented brilliant, groundbreaking work. Strictly, it would be a stretch to consider the Difference Engines to be more than an elaborate calculator—an incredible achievement for the day, to be sure, but a calculator nonetheless. They were much closer in nature to a mechanical clock than to a computer.

The Analytical Engine was a much more sophisticated device. The Analytical Engine is much more than a calculator and marks the progression from the mechanized arithmetic of calculation to something that was the mechanical precursor to the modern computer. It was programmable using punched cards. The engine had a "store" where numbers and intermediate results could be held and a separate "mill" where the arithmetic processing was performed. It was also capable of functions used in modern computing systems, like conditional branching, looping (iteration), microprogramming, parallel processing, iteration, latching, and detection and handling of errors (Fig. 2.2).

It was during World War II that the first modern electronic data processing computers, known as Colossus, were developed. Little is known about the machines because they were developed and

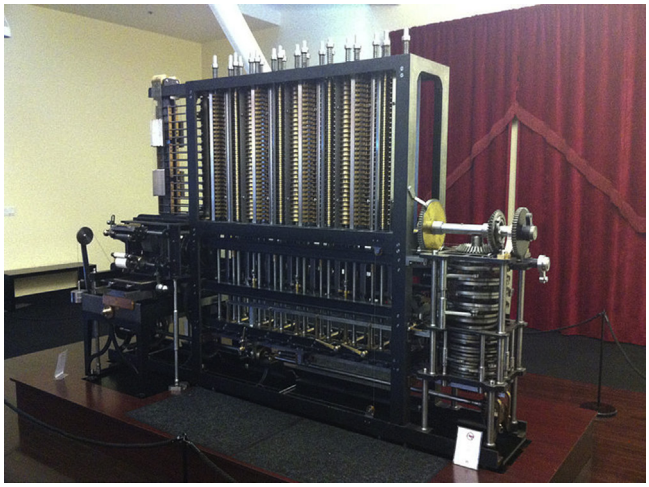

FIGURE 2.2

Analytical Engine.

successfully used by Britain to decrypt coded German messages. These machines were considered ultra secret and were so effective that they continued to be used during the Cold War. Because of the secrecy surrounding them and the very effectiveness of the devices, when they were superseded by more powerful equipment, these early electronic computers were quietly and secretly destroyed.

The next major advance in computing came with the creation of ENIAC in 1946. It was programmed with patch cables and switches. Patch cables continued as a form of programming into the early 1970s. The next advance was the replacement of relay switches with vacuum tubes and later with transistors. Another advance was the introduction of random access storage devices. Programs moved from patch cables/boards to paper tape and then to punch cards.

In the late 1960s and early 1970s, programs began to be referred to as applications. Individual programs were run in a series of steps. Initially, these pieces were loaded (run) manually. Eventually, they came to be connected via **Job Control Language (JCL)**. These interconnected programs became application systems such as payroll, accounts receivable, general ledger, etc. Also in the 1970s, online systems and computer networks made their appearance.

DEFINITION

Before proceeding any further, it is important to establish a common understanding of the term **application management**. In the broadest sense, an application may be considered any software or any program that runs on a computer. It may be "firmware"—that is, software that is written on a chip. It may be the **operating system (OS)** or associated utilities. However, in general, the term **application** refers to those collections of programs that are designed to perform a particular function that entails the processing of data and subsequently generating some form of output.

Application management consists of the monitoring and control of application software with objective of optimizing its performance, availability, and security in order to meet service-level commitments.

In the world of information technology (IT), to manage something means to monitor it and to control it. Therefore, application management means the monitoring and control of application software with the objective of optimizing its **performance** and availability in order to meet service-level commitments. That management may be done by humans or by other software. The objective is to optimize the performance, availability, and **security** of the application. This is discussed in greater depth in Chapter 3, "Managing Traditional Applications."

Management is always an afterthought.

THE EARLY DAYS

This is a good time to lay out a universal, but unfortunate, principle of management: *management is always an afterthought*. It does not matter what the technology is. The innovators initially rush to build, prototype, and deploy. Only after some initial success is the need for management and instrumentation (i.e., the ability to monitor and to measure) recognized and addressed. The need may be recognized by the creator/builder/engineer or it may be the user of the technology who demands that manageability be incorporated into the product.

Consider the Wright Brothers. On December 17, 1903, near Kitty Hawk, North Carolina, Orville Wright made the first powered flight in the Wright Flyer. This was only a little more than a glider with an engine strapped to it, but it was revolutionary. Management of the aircraft was almost nonexistent. The only instrument included was a handheld anemometer that was taped to a wing strut. It did not have any of the instruments that became quickly standard. However, it did not take long for pilots and designers to realize that there were some basic components that were critical, including a tachometer, altimeter, compass, fuel indicator (later a gauge was introduced), airspeed indicator, bank or turn indicator, climb indicator, oil pressure gauge, and oil temperature gauge.

Like the Wright brothers' early aircraft, applications and the management of them were equally primitive in the beginning. In the early years of computing, computer pioneer Grace Hopper (Fig. 2.3) coined the term "bug" to refer to system (including program) malfunctions. Moths, attracted to the heat of a computer, would crawl inside of a computer. Once inside, a moth had the potential to interfere with the operation of one of the relay switches, which was how program instructions were executed. One time, on finding a moth causing of malfunction of a relay switch, Ms. Hopper declared that she had found the "bug," thus coining the now ubiquitous term. Moths could also get stuck on a paper tape containing a program or data. This would prevent the holes in the tape from being detected, thereby altering the program or the data that were encoded on the tape.

Also like the Wright brothers, the people leading the way in computing learned by trial and error. They waited for something to break, tried to figure out why it had happened, and then tried to fix it and find a way to prevent it from happening again. Therefore, in the late 1940s and very early 1950s, application management largely consisted of waiting for something to break and then fixing it. At that time, the primary cause of failures was the complex array of hardware components (particularly vacuum tubes, relay switches, and complex circuitry). If there was a problem, the immediate and natural

FIGURE 2.3

Grace Hopper and team at the console of an early UNIVAC mainframe computer.

assumption was that the cause would be found somewhere in the hardware. Only after exhausting the hardware possibilities was the software considered as the possible explanation. Tracking an application's progress was relatively a straightforward thing to do since only one program could run at a time.

THE 1960s

Fast forward to the 1960s. Led by the introduction of IBM's System 360 and with it OS/MFT, operating systems of computers had matured to the point that applications were no longer single-threaded (i.e., only one application could run at the same time on a single computer). Now it was possible to run multiple applications at the same time on one computer. These applications ran in separate partitions of the computer's memory, isolated from each other. Data sharing between applications was generally done through files rather than through a direct exchange of data between the applications. While multiple applications could run on a single computer, those applications still ran in batch mode. That is, an application would be started, process the data it was given, and then terminate (i.e., shut down). In the 1960s, online, real-time systems were still the stuff of fantasy and research laboratories focused on the most advanced technologies.

By the 1960s, the hardware had also matured; it was much more stable and reliable. True, computer systems were still large and ran in special air-conditioned rooms, with special air filters and "conditioned power" (i.e., run through special equipment to minimize fluctuations in voltage or amperage); in some cases, chilled water was required for cooling the computer(s). If there were hardware problems, the OS could often give some indication of the nature of the problem and where it occurred (Fig. 2.4).

FIGURE 2.4

System 360.

Few products in American history have had the massive impact of the IBM® System/360—on technology, on the way the world works or on the organization that created them. Jim Collins, author of Good to Great*, ranks the S/360 as one of the all-time top three business accomplishments, along with Ford's Model T and Boeing's first jetliner, the 707. It set IBM on a path to dominate the computer industry for the following 20 years.*

Most significantly, the S/360 ushered in an era of computer compatibility—for the first time, allowing machines across a product line to work with each other. In fact, it marked a turning point in the emerging field of information science and the understanding of complex systems. After the S/360, we no longer talked about automating particular tasks with 'computers.' Now, we talked about managing complex processes through 'computer systems.'

At the time, however, success was far from clear. Thomas Watson Jr.'s decision to pursue this vast, US$5 billion investment in something that would cannibalize the company's existing product lines was an epic 'bet the business' move—one that emerged as much from Watson's determination to prove he could live up to the legacy of his father, IBM's legendary founder Thomas Watson Sr., as from changes in technology. Within IBM, the S/360 project sparked an extraordinary period of technological and business creativity, internal conflict and self-reflection for thousands of IBMers. When the S/360 was announced on April 7, 1964, it not only changed computing forever, but also IBM. The company learned, in Watson Jr.'s words, that 'there was nothing IBM couldn't do.'

The S/360 replaced all five of IBM's computer product lines with one strictly compatible family, using a new architecture that pioneered the eight-bit byte still in use on every computer today. The announcement was revolutionary in concept and unprecedented in scope. Six processor models were announced, covering a fifty-fold range in performance, as well as 54 different peripheral devices.

The System/360 announcement changed the way customers thought about computer hardware. Companies for the first time could buy a small system and add to it as they grew. Companies other than IBM found they could make peripheral equipment that worked with the S/360. An entire industry was soon created, consisting of companies making and supplying plug-compatible peripheral products….

In 1989, a quarter of a century after IBM System/360 debuted, products based on S/360 architecture and its extensions accounted for more than half of IBM's total revenues. They also accounted for more than 50 percent of the US$260 billion worldwide inventory of all computers made by all companies and priced over US$100,000 each.

http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/system360/

A **master console operator (MCO)** would sit at a terminal and watch the console for signs of problems with hardware or software. The MCO would manually start applications by loading punch cards containing the program into a card reader. At that point, the execution of the application would either be controlled by the JCL commands or manually by the MCO. The MCO would also watch the progress of applications and their completion. The MCO would receive a notification when an application required some action to be taken (e.g., mount a specific magnetic tape, or a particular disk pack, etc.). If an application failed, an error message would usually be given to indicate the nature of the nature of the problem that caused the failure. In some specific cases, the MCO could resolve the problem; however, the MCO had to contact the application programmer to troubleshoot and resolve the problem. Often, if a job (i.e., the running of an application with a specific set of data) failed, after the problem was resolved, it would be necessary to run the entire job again—from the very beginning. With jobs that took a long time to run or ran on computers that had little or no extra capacity, this could be a serious problem.

Some applications were very large, with instruction sets of thousands of punch cards. While today that may seem trivial, an important management feature consisted of punching a sequence number on each of the program's cards. That was a safety measure so that when, sooner or later, someone inevitably dropped some (or all) of the cards, they could be quickly resorted into the correct order.

In short, in the 1960s and into the early 1970s, application management was still largely a manual process and that was becoming more complex and difficult. Thinking back to the example of aviation, the progress of application management in the 1960s was comparable to the early instrumentation and navigation aids available to pilots by the outbreak of World War I. By that time, pilots had the bare minimum required in terms of instrumentation and navigation aids (a compass and a window). Of course, the airframes were better and stronger, the engines were infinitely more powerful and reliable, and speeds had increased dramatically. All of this is surprisingly similar to how the computing environment was evolving.

THE 1970s

It was in this decade that computing really began to mature. No longer was the use of computers limited to just the largest companies or government agencies needing to process large amounts of data. As the cost of computer components followed Moore's Law, prices continued to fall and computers became affordable to more and more companies. Also during this period, computer systems continued to become faster and more complex and, with that, the management of the applications became more difficult.

> **"Moore's Law"** states that the number of transistors in an integrated circuit doubles every 24 months. This concept was introduced by Gordon E. Moore, co-founder of Intel.

Developers of the OSs of that era sought to provide some relief for the management problem with some basic functionality incorporated into those systems. They improved the ability of the OS to recognize the cause of an application failure and report it when the job terminated. The OS also tracked resource utilization by each application. The information was useful for job **accounting**, charge-back

systems, and capacity planning. The latter was the analysis done to determine when it was necessary to purchase a new computer or upgrade an existing one. Reports were developed from this information to permit system programmers, application programmers, and MCOs to study the utilization of the **central processing unit (CPU)**, I/O (input/output) channel and storage (disk and tape), and how long it took for programs to run. This data could be used to determine if programs were running efficiently. Those that appeared to be inefficient or had high resource usage ("resource hogs") would be analyzed to determine what changes could be made to the programs (e.g., processing techniques, file structures, etc.) to make them more efficient. Small changes could sometimes lead to a significant reduction run times, resource consumption, or the overall system load.

It was in the 1970s that online, real-time processing began to be become mainstream. Initially it was a novelty, with terminals that were similar to an electric typewriter. These devices were universally slow. Their speed was constrained by the speed of the connection to the mainframe and even more by the speed at which they could print the output. In 1971, IBM introduced the 3270 CRT terminal (often referred to as a "green screen" because of the color of the characters on the original models). These terminals were attached, through a control unit, to a mainframe (Fig. 2.5).

Online, real-time processing meant that users could have immediate feedback. Agents at airline ticket counters, bank tellers, or even data entry clerks could have immediate access to information or get immediate feedback about their input. Similarly, the data entered by these people could be immediately entered into corporate files (if the business chose to do that). For application programmers in

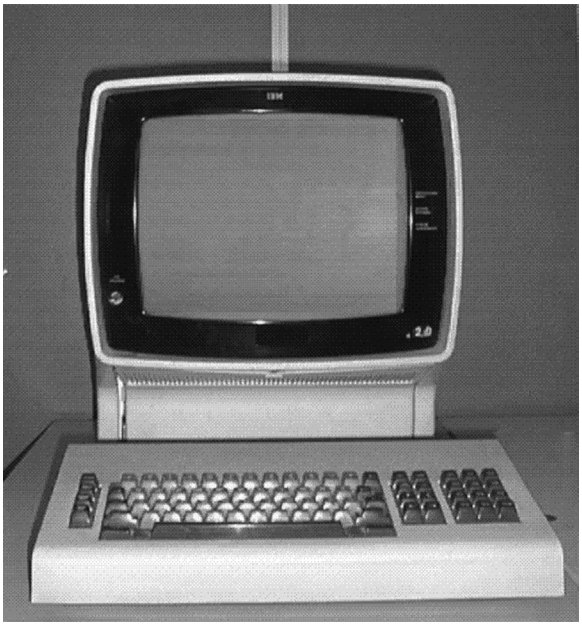

FIGURE 2.5

3270 Terminal.

particular, this was a paradigm shift. Gone were the days of submitting test jobs and then waiting, perhaps 24 hours, to get the results. Now they could have very quick turnaround for their test jobs, provided that resources were available to allow the test jobs to run.

The arrival of online, real-time processing was the tipping point for application management in **mainframe computers**. It was no longer acceptable for resolving an application failure to take several hours. Now, users were interacting with the applications on a real-time basis and businesses required that availability and interaction in order to be able to function. Downtime (i.e., system outages) was expensive: revenue could be lost and/or unnecessary expenses could be incurred because staff were unable to do work when scheduled. IT turned to automation to improve the management of applications. Larger organizations created some programs to automate certain critical functions. However, most turned to their system supplier (which was increasingly IBM) and to third-party software developers.

It was also in the 1970s that the minicomputer (e.g., IBM System 32, DEC VAX, etc.) made its debut. Although minicomputers generally used a different OS than their larger, mainframe cousins, in terms of application management, there were no significant differences.

Some of the early application management functions to be automated are shown in Table 2.1.

The IT world was not and is not perfect. It never will be. It never can be. There were still unplanned application outages and some of them were protracted. However, overall, both performance and availability of applications improved significantly over what they had been in the 1970s. Some of that could be attributed to the automation of application management and some was attributable to improved hardware reliability.

Minicomputers appeared on the IT scene in the 1970s. To borrow from Yogi Berra, it was "Déja vu all over again." There was a stampede to deploy this new technology without sufficient thought given to how it would be managed. IT had to work through the same problems that it had faced in the mainframe environment.

Table 2.1 Examples of Early Application Management Tools

Role	Description
Tape management	Mapped the location of files to specific reels of magnetic tape and allowed the application to request that a specific reel be mounted on a tape drive.
Data backup and recovery	This relieved that application programmer from having to write code to ensure that data used by the application would be regularly backed up. It also greatly simplified the process of restoring a data set.
Job scheduling	This was intended primarily for applications that were run in batch mode. It allowed someone in operations to define the sequence in which jobs would be run. It also allowed dependencies on other resources to be defined. Job scheduling was also useful for ensuring that online systems were started at the required time. It eliminated the problem of an operator forgetting to start the online system.
Checkpoint restart	This tool would monitor an application. If the application failed or if the process was otherwise interrupted, this tool could be used to restart the application at the point in its processing where it was when it failed.

THE 1980s

Within the data center, automation of application management functions continued throughout the 1980s. Increasingly, IT departments turned to automation tools sold by third parties, also known as **independent software vendors (ISVs)**, rather than by IBM (whose hardware technology, by the late 1980s, was the de facto standard for mainframes). Top priority was given to tools that would monitor **application performance** and availability.

With the 1980s came the adoption of the **personal computer (PC)** for business uses. Initially, they were dismissed by IT departments as not being "real computers." The early ones had only 64 KB of memory and were prone to frequent crashes. Data and applications were stored on and loaded from 5¼″ floppy diskettes that held a mere 110 KB of data. In fact, those early PCs were relatively expensive when their limited functionality is taken into account, especially comparing the price and capabilities of today's PCs. They were capable of limited word processing, running basic spreadsheets, and playing games. In short, they were fine for a hobbyist but not for doing serious work.

However, the PC began to mature. As they matured, businesses began to adopt them. Initially, the use of PCs was confined to small, specialized tasks within a department. As **line of business (LOB)** managers saw what PCs could do for them and that it could be done without incurring the charge-backs for IT assessed by the accounting department, with who-knows-what included (floor space? part of the break area? etc.), they continued to proliferate. Next came the demand to connect the PCs to a mainframe or minicomputer. Once connected to the mainframe, they acted as smart terminals, capable of pushing data to the mainframe as well as receiving and storing information.

In terms of application management, PCs bore a very strong resemblance to mainframes of the 1960s. Since there were no management tools, if you knew how to get to it and interpret it (most users did not know), there was some management information available from the operating system. However, it was very limited and arcane. The user filled the roles of both MCO and the application user. Again, like the mainframe operators in the 1960s, the user would run their application (just one at a time) and pray that it would not crash. If the application did fail, they would either have to figure out why it did and fix the problem, or find someone willing and able to help them.

An application management issue of particular concern that arose as the use of PCs became widespread was that of software distribution/version control. It became important to make sure that PCs scattered across the enterprise were all running the same version of each application that interfaced with the mainframe. It was a problem that was never completely solved and still it persists with PCs and servers in the current **distributed computing** environment.

THE 1990s

The mainframe continued its evolution in the 1990s, as pundits around the world proclaimed its imminent demise. Tools for application management became more sophisticated and powerful. The use of them grew and provided benefits in the form of improved availability and performance. Similarly, PCs became ubiquitous in the enterprise. However, the application management issues faced on the PC were more intractable.

In the 1990s, two "elephants" that pushed their way into the room were the Internet and client/server computing. At first glance, it does not seem like the Internet should have had a significant impact

on application management. It did not have to have a material impact and it would not have, if organizations were content run corporate "intranets" and not connect the enterprise to the global Internet. Unfortunately, the temptation to grab that shiny new toy was too great to resist. Organizations, first in the United States, and gradually around the globe, started to connect their private networks to the public Internet. Usually, this was done with little regard to whether there would be material benefits from doing so. There was no risk-versus-reward assessment. It was a shiny new toy and every self-respecting technologist has to have all of the latest toys. It was *really cool* and what could possibly go wrong? (A lot!) Furthermore, it held the promise of saving money by eliminating some of an enterprise's private network expense (the same technology that enabled the Internet also enabled the deployment of corporate **local area networks (LANs)** and **wide area networks (WANs)**).

Each organization that connected their infrastructure to the Internet opened Pandora's box. The problems were not immediately apparent. It took time, but gradually the consequences started to be exposed. As the hackers started to knock on the "door," it became obvious that applications had to be better protected. Certainly network security was required, but that is not truly application management. The applications needed to have a higher level of security than the application developers were incorporating into their applications. Ever since then, there has been an ongoing struggle between the hackers and those trying to defend the corporate assets. Short of businesses disconnecting from the Internet, this struggle is going to continue indefinitely.

With almost any new technology, there is a lag between the time innovation occurs and the widespread adoption of it. With client/server, widespread adoption really took off in the 1990s. When compared by most metrics to a mainframe, those servers were cheap, *really* cheap. Like the adoption of PCs in the 1980s, the servers were cheap but did not come with management tools. It was another case of "back to the future."

Similarly, the adoption of new networking technologies for similar reasons brought comparable management problems. Of course, by this time there was a burgeoning community of third-party software developers that were ready to charge in to the gap to fill the management void in both the server and the network domains. That included addressing the full set of application management functions. By the end of the decade, developers in Silicon Valley and around the world had responded to the needs created by these new technologies, and application management in the client/server was able to be as robust as what was being done in a mainframe environment. In some cases, it was even more advanced than on the mainframes.

Together, innovations in networking, **client/server architecture**, and distributed computing led to demand from clients for the vendors of products used in these areas to support open, standards-based management. That is, the users wanted to be able to decide which management tools they would use to manage any part of the IT infrastructure. For example, they did not want to be forced to use a specific tool from the manufacturer of a router to manage those routers. The significance of this for application management was that it led to a drive to enable integrated management of the entire IT infrastructure. Just as an application did not exist or run in the absence of a computer, it could not be managed while ignoring the infrastructure that enabled it. This became even more important in the 21st century.

It may seem like the emergence of client/server, Internet, and distributed computing were just one more step along the path in the evolutionary journey of information technology. In a sense, that is true. However, taken together, they represented something much greater—a watershed period for the changes that were to follow. They opened the door for the explosive changes that have taken place since then.

Cloud computing, **virtualization**, **componentization**, etc., can all trace their routes to those changes that took place in the 1990s.

THE 21ST CENTURY: THE FIRST DECADE

The period of 2000–09 saw dramatic changes in how applications were built, how they were deployed, where and how they were run, and how they were managed. The discussion of the advances in this decade and the next will be brief. That is because they will be discussed in much greater depth in later chapters of this book. With that caveat, we can begin a quick review of the developments of the 21st century.

This was a period of changes that were simultaneously revolutionary and evolutionary. It was during this period that **service-oriented architecture (SOA)** began to be widely adopted. It meant that an application was now seen as being comprised of components, each providing a **service** when invoked by another component or by a user. It standardized the exchange of data between components and exponentially increased the complexity of applications and the management of them. Software companies in the business of building management tools were somewhat slow to respond to this demands of this new paradigm. Initial attempts were expensive, complex, labor-intensive to implement, and difficult to use. The ability of IT departments to manage applications actually regressed during this period.

Cloud computing comes in three forms: private, public, and hybrid. **Private clouds** do not have a material impact on application management. The applications are still running on infrastructure that is owned and operated within the enterprise. Therefore, it is possible to use the same management tools and disciplines as are used on other systems. **Public clouds** can create new challenges for the management of applications. Whether or not a public cloud presents those challenges really depends on the nature of the cloud and the relationship between the cloud provider and the client. If the "cloud" is really just a **platform as a service (PaaS)** (i.e., the client is contracting for exclusive use of a computer and other resources), then this is essentially the same as a private cloud. The **hybrid cloud** reflects a situation in which organizations can be thought of as having a foot in both worlds. That is, hybrid reflects the use of a combination of both public and private cloud.

It is when the "cloud" entails multiple enterprises running applications on the same computer that management challenges arise. In that type of environment, actions that a client can take are limited and tightly controlled. The client can see whatever management information is provided by the application itself. Anything beyond that is a matter of the policies of the cloud service and the agreement with the client. Similarly, the corrective actions that the client can take are generally very limited.

Virtualization is an essential technology that is the underpinning of cloud computing. It introduces a significant increase in the complexity of the environment in which applications run. However, it does not directly impact the management of applications.

Software as a service (SaaS) really had its roots in the dot-com bubble of the late 1990s. Survivors of the crash that followed the bubble learned important lessons. They proceeded to build companies to offer the use of an application software solution. One of the more famous and successful SaaS companies is SalesForce.com. From the perspective of application management, the SaaS environment is a problem. The client is not able to instrument the application or the system on which it runs. In short, the client is not able to manage SaaS applications. The decision to use SaaS applications has to be a risk-versus-reward tradeoff.

Mobile applications also appeared in this decade. However, initially, the apps on them largely targeted consumers. For that reason, we will wait to discuss them in the next section.

THE TEENS (2010—PRESENT)

As we begin our look at the last period, it is worth noting that the demarcation between this period and the preceding one is not as sharp as we perceive in those between other decades. Part of that is because we are still living through it. Also, much of what is happening reflects the continuation of changes begun in the previous decade.

Today, SOA is the norm. Everything is componentized and parts can be reused (management of componentized applications will be addressed in Chapter 9). Applications morphed from being encapsulated entities in a single, unified environment that once was the option. Now, applications are spilling across the enterprise and out of the enterprise via **application programming interfaces (APIs)** into a fabric that is forming and reforming again, bonding with various components as needed to satisfy a particular service call. Applications have become dynamic and capable of constant change, like a virulent virus.

Because of that dynamism, we are faced with the challenge of simultaneously managing each component as a discrete piece, plus managing all of the components involved in processing a transaction as a single entity. It requires a holistic view that is not limited to just the application software as the "managed object." It is necessary to take into account the resources that an application uses (e.g., systems, networks, databases, etc.). Each of those resources has the potential to impact the performance of an application, while masquerading as a problem with the application.

You may be able to look at a single component and, if you are lucky, discover that it is the cause of the problem that you are addressing. However, the odds are against you. The number of pieces has soared exponentially, and the cause of a problem may not even lie with a single piece but rather the interaction between the pieces.

Manual management of such a complex and fluid environment is not a reasonable strategy. Fortunately, tools for application management have become very intelligent and much easier to use. So far in this decade, there has been an exponential leap in the capabilities of tools for application management. For example, one application management tool (AppDynamics) is able to provide up to the minute real-time **topology** mapping (of properly instrumented applications). Autonomic application management is becoming a reality.

Despite the growing body of powerful tools for application management, research by Enterprise Management Associates (EMA) repeatedly found that less than half of enterprises are actually using tools intended specifically for application management. How can this be reconciled with the statement above that manual management of modern, dynamic applications is not a reasonable strategy? The answer is twofold. First, many organizations have not made the transition to these modern apps or have only done so on a very limited basis. Traditional applications are the mainstay for those organizations. Management of those applications will be discussed in Chapter 3, "Management of Traditional Applications." The other half of the answer is that some organizations have deployed modern applications, but are still relying on the old "break/fix" approach to management of them—a very shortsighted approach that is certain to lead to missed service-level commitments and disgruntled users.

There two other major changes in this decade: mobile applications and the **Internet of Things (IoT)**. Particular challenges in each are connectivity, security, and massive scale. Mobile will be discussed in Chapter 6, "Management of Mobile Applications," and IoT is discussed in Chapter 15, "Application Management in the Internet of Things."

SUMMARY

The history and development of application management have been inexorably tied to the evolution the entire spectrum of the associated technology (networks, computer systems, operating system software, etc.). While tightly coupled with the other parts of the IT infrastructure, management technologies (including application management) have always lagged behind the other areas of innovation and evolution. Table 2.2 summarizes the developments discussed in this chapter.

Table 2.2 The Evolution of Application Management

Period	Key IT Technology Advances	Application Management
1950–59	Commercial adoption of computers	Application management happened in a reactionary mode without the aid of tools.
1960–69	Mainframe computers	Error logs were the primary tool for application management.
1970–79	Online, real-time processing	Rudimentary information for application management began to be provided, primarily by the operating systems.
1980–89	Evolution of mainframes Appearance of minicomputers Personal computers	Growing body of ISV management tools for mainframe applications, less for minicomputers. Virtually no application management resources were available for PCs.
1990–99	Internet/LANs/WANs Client/server architecture Distributed computing	Open, standards-based management tools, hardware and software. An integrated, holistic approach to application management.
2000–09	Cloud computing Virtualization SaaS	Applications no longer run in a single, unified environment. Management capabilities regressed relative to the new environments. SaaS offers lower cost application functionality, but takes away from the enterprise the ability to manage those applications.
2010–present	SOA Componentized applications Mobile applications IoT	Sophisticated, automated management tools have emerged to help manage this dynamic environment. Mobile applications present challenges to application management in terms of communication, security and scale.