CHAPTER

DISTRIBUTED AND COMPONENTIZED APPLICATIONS

9

All things are difficult before they are easy.
Chinese proverb

INTRODUCTION

The growth of distributed **applications**—which can include everything from **web services** and **service-oriented architecture (SOA)** services to applications leveraging **application programming interface (API)** connections—introduces organizational, technical, and governance challenges. The applications themselves run across silos, and the skills and tools required to manage them are far more cross-functional than those required to support, for example, Microsoft Word running on a desktop.

Payment processing, airline reservations, and high-speed trading are examples of large-scale componentized applications. In each, transactions execute across front-end systems of engagement and back-end systems of record. They may traverse a variety of databases, along with **content delivery networks (CDNs)**, web infrastructure, and devices. In many cases, these transactions also execute across third-party systems such as those provided by payment card providers, brokerage networks, and credit reporting agencies.

While a desktop application is encapsulated on a single device, componentized applications are almost unimaginable in their complexity. They can run across thousands of devices and hundreds of diverse platforms and code bases, traverse the Internet, and access data from the data centers of partners or suppliers. At the same time, these are among the most business-critical applications on the planet, with exceedingly high requirements for performance and availability. The question is, how does this combination of complexity and business criticality impact **application management**?

The intent of this chapter is to provide a background and foundation for the following three chapters:

- Chapter 10 covers "**DevOps** and **Continuous Delivery**." DevOps provides the cross-functional **management** capabilities that are fundamental foundations for application support. DevOps also provides a dynamic core supporting continuous delivery, the processes relating to delivering new software into production on an ongoing basis.
- Chapter 11, titled "**Application Programming Interfaces** and **Connected Systems**," details the management challenges relating to providing and consuming APIs and to delivering **container-based microservices**.
- Chapter 12 on "**Application Performance Management** and **User Experience Management**" covers the gamut of **application performance management (APM)** and **user experience management (UEM)** solution types and use cases.

Application Performance Management (APM) in the Digital Enterprise. http://dx.doi.org/10.1016/B978-0-12-804018-8.00009-7

This chapter and the next three discuss and elaborate on the theme of managing complex applications, each providing specific detail about the process- and tools-related aspects of a particular dimension of application delivery.

While distributed and **component-based applications** present a wide variety of challenges to virtually every information technology (IT) department, they also compose the majority of applications supporting virtually every mid-sized to enterprise-sized company. Developing an understanding of the elements that go into managing these types of applications is a necessary foundation for successful tools planning.

APPLICATION DIVERSITY

Although the consumer user typically thinks of an "application" as a mobile app or a mail program such as Microsoft Outlook, the majority of the applications running in today's mid-sized and enterprise-sized data centers are far more complex and less straightforward in function. Fig. 9.1 shows the wide diversity of component-based and **integrated applications** delivered by today's companies. It also shows that a large percentage of companies are running these types of applications.

As Fig. 9.1 also shows, many of these application types interoperate with other systems. Web services, typically understood to be **representational state transfer (REST)**- or **simple object access protocol (SOAP)**-based services, run in about 50% of companies. **Integrations** with mainframes—typically via APIs or web services—are the second most common type of integration. **Enterprise**

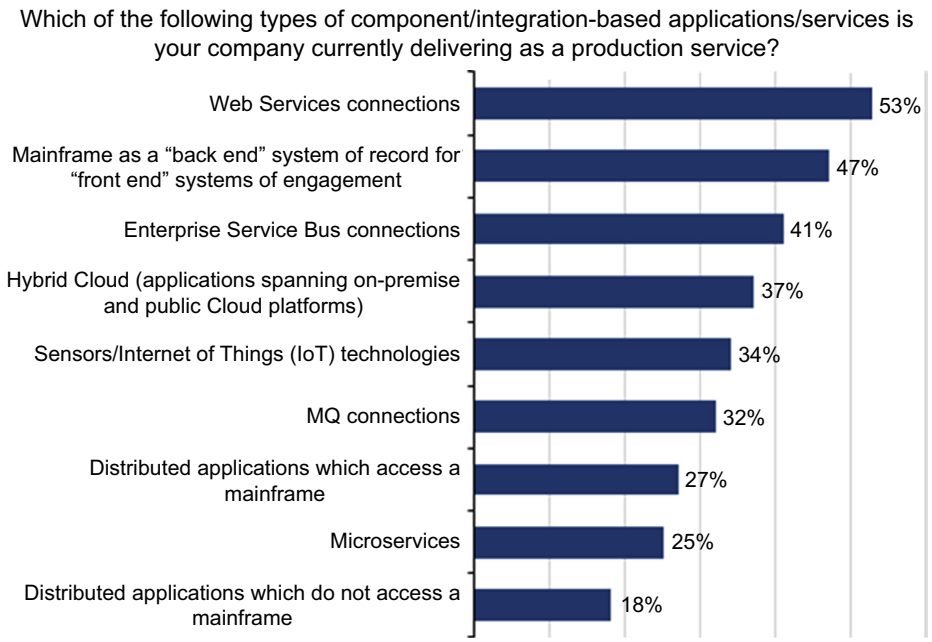

FIGURE 9.1

Integration technologies used for production service delivery.

service bus (ESB) connections, which are typically used to integrate internally hosted applications, run in about 40% of companies.

Even a "simple" banking app running on a mobile phone may be supported by a wireless network, web-related applications and servers, mainframe systems of record, databases, and a host of other infrastructure elements. While a user views an app as an icon on a phone, IT support teams view the same app as a transaction executing across a wide array of complex infrastructure. A glitch at any point in the execution chain impacts the end user in the form of slow performance or even an outright failure to execute.

The transactions spawned by these applications execute across software and data scattered across a broad array of diverse platforms. They often run across distributed servers running on heterogenous platforms, **operating systems**, and programming languages. They may access code from multiple applications and data from multiple databases. They are highly network-centric and may even execute in part in the **cloud** or on other systems external to the corporate **firewall**, such as partner or supplier systems.

And while many of these applications are built on traditional platforms, new application **development** often focuses on technologies such as APIs, containers, and even **Internet of Things (IoT)** applications. Modern development teams are using **agile** development practices. This, in turn, has led to the growth of continuous delivery as the primary methodology for delivering new software applications, features, and functions.

Fig. 9.2 shows the types of applications most frequently delivered via continuous delivery practices—basically the "new" application types that are now being deployed to production. One takeaway

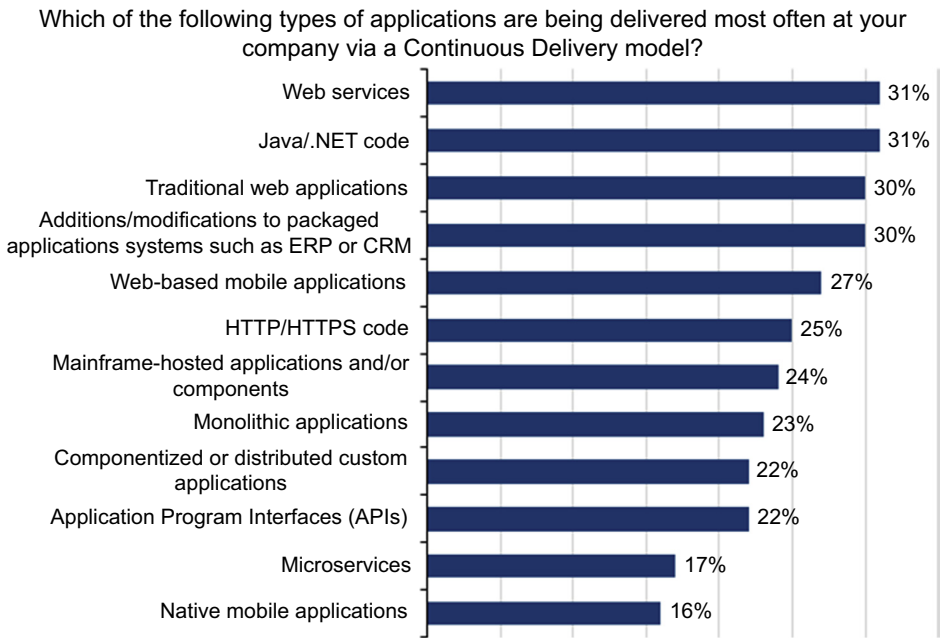

FIGURE 9.2

Application types most frequently developed/delivered via continuous delivery practices.

from this graphic is that, of the types of applications that are currently in development, almost all can be considered "distributed." With the exception of "native mobile" and "monolithic applications," every application type in this chart can legitimately bear this description.

In short, managing the heterogeneity underlying today's enterprise applications presents significant challenges in the areas of staffing, tooling, performance, and availability. Windows, mainframe applications, and databases coexist with .NET, Java, and COBOL components, and this diversity creates risk. Each platform is typically supported by its own team of specialists relying on siloed management tools. Few companies have accurate, comprehensive **topology** models of composite applications and fewer still have the "industrial-strength" APM solutions necessary to automate the support process to any meaningful degree.

THE EVOLUTION OF APPLICATION COMPLEXITY

Much of this complexity evolved over time. Fig. 9.3 shows the evolution of technology and application architectures since the introduction of **client/server architectures** in the mid-1990s. In the "distributed era" users accessed applications that were generally delivered by private data centers.

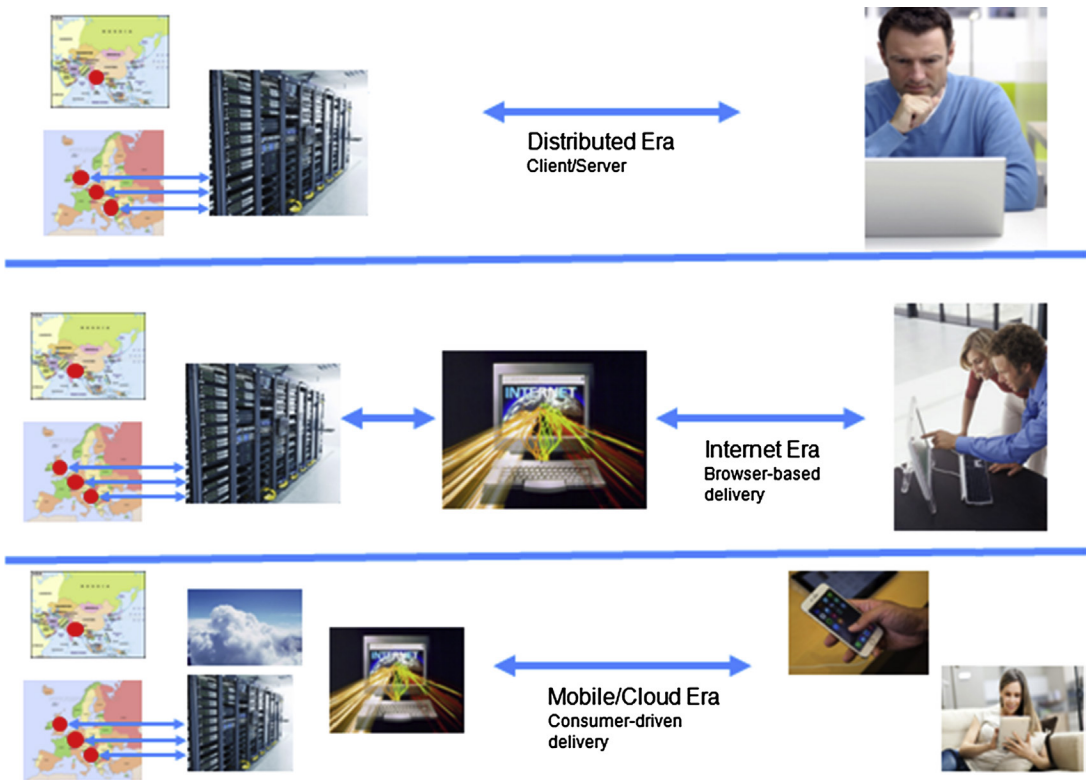

FIGURE 9.3

Evolution of distributed computing.

As the Internet matured and companies began delivering web applications to employees and customers in the "Internet era," web-enabled applications became the norm. Finally, the "mobile/cloud era" depicts the age in which "consumers" became "end users," when Internet-connected applications became ubiquitous, and when companies of every size were scrambling to web-enable traditional applications and consumer-enable new ones. This is also the era in which transactions routinely exit private data centers and execute, at least in part, in the cloud or on third party systems.

It is interesting to note that each era includes additional elements and more connection points than the preceding one. It is also interesting to note that the promise of the Internet to connect "anything to anything" was indeed achieved. Consumers empowered by easy access to the Internet and a host of high-performing **mobile devices** ushered in the "era of the consumer user." As a result, both on-premises and cloud-hosted applications not only became increasingly complex, but also became increasingly business-critical revenue drivers.

At the same time, while users expect to access applications as easily on tablets and phones as they do on laptops, they have also become accustomed to the sub-second performance that was the norm during the distributed and early Internet eras. And while IT professionals understand that it is reasonable to assume that a transaction traversing multiple geographic locations takes longer to execute than one that is direct-connected to a backend system, users do not see it that way. They have no knowledge of application architectures or locations or network speeds, but rather judge application performance from the perspective of their own subjective experience. These realities—combined with the dollar value attached to virtually every "consumer user" in this mobile era—are key reasons for the growth of both APM and UEM solutions.

Applications built over SOA and/or container-based microservices usher in a new concept in application delivery. SOA components and microservices are developed as "building blocks." They provide examples of services (versus applications) that are built by orchestrating the execution of multiple software components; further, these components can be internal or external to a company.

These componentized services introduce an additional concept into the "distributed" definition, which is the concept of integrations. SOA components can be connected via ESBs, SOAP, or REST. Over the past 5 years, SOA architectures became the norm for both new and refactored applications. They have become increasingly important for restructuring and modernizing traditional application portfolios or, at the other end of the spectrum, acting as the predominant structural platform for new software services. While approximately 60% of companies indicate that they are at some stage of SOA adoption, to some degree many still rely on homegrown management products to manage these complex, componentized services.

By definition, microservices, the latest generation of component-based services, are virtually always connected via APIs. Today, microservices hosted in containers such as **Docker** are adapting the orchestration concept to a new breed of uber-connected, uber-integrated services.[1] Again, this new functionality requires updates adding API and container visibility on the part of APM vendors.

[1]Chapter 11, titled "Application Programming Interfaces and Connected Systems," describes API and container concepts in detail.

HETEROGENEITY, SCALE, AND INTEGRATIONS: THE "LOOSE CANNONS" OF APPLICATION PERFORMANCE

Three primary characteristics make "complex applications" complex: heterogeneity, scale, and integration. Each element adds unique challenges to the task of application support.

HETEROGENEITY

Supporting heterogenous applications and platforms requires both tools and in-house skills addressing each silo platform. For example, when applications are running on Windows, Linux, and z/OS as examples of diverse operating systems, an IT organization must invest in personnel and tools capable of supporting each of these areas of expertise.

It is important to bear in mind that the new technologies introduced in each of the different eras did not replace existing systems. Instead, each technology was added to the growing inventory of hardware and software that already existed in virtually every enterprise data center.

Today, most larger companies are supporting modern applications and infrastructure alongside applications running on traditional platforms and using older protocols. "Don't fix what ain't broke" may be the golden rule of IT, and many IT professionals report that their data centers contain older devices that are still running—even though nobody can remember which applications or users they actually support.

One reason for this is that both applications and data have become valuable business assets performing critical business functions. The 15-year-old server sitting in the corner may well contain data or spreadsheets used by the CFO for financial or compliance-related reporting.

There is often business risk associated with retiring old infrastructure and introducing new hardware and software into production. Most enterprise IT organizations learned over time that it is far less risky to continue to build on proven systems than it is to retire those systems and modernize via hardware purchases and software rewrites. A great deal of traditional mainframe software, for example, was componentized and deployed first as SOA services, then "wrapped" in web services protocols such as SOAP and REST. These practices extend the functional lifetime of software assets, while also modernizing them for access by APIs, ESBs, and similar integration technologies.

SCALE

Scale is another growing problem compounding the challenges associated with application delivery. And because both heterogeneity and complexity are two key factors driving scale, this factor is very difficult to mitigate. Scale is an issue for a variety of reasons. However, the primary reason is based on a simple equation:

More Components = More Potential Failure Points.

Today's IT organizations, particularly those in enterprise-sized companies and telecommunications carrier environments, are managing a level of scale that can be difficult to conceptualize for users whose experience is limited to laptops and mobile devices. A single **mobile application** may be

supported by hundreds or thousands of hardware and software elements. And that is just one of the hundreds of applications running in the average enterprise-sized company. EMA's latest research found that most mid-sized companies are running between 50 and 100 production applications—and that approximately 10% (typically larger companies) are running more than 1000.

The impact of **virtualization** and, more recently, container-based microservices is another element of scale. In either type of environment, particularly when such technologies are hosted on cloud-based **infrastructure as a service (IaaS)**, scaling becomes a relatively simple matter of duplicating a container, a **virtual machine**, or a server cluster. So from the APM and UEM perspectives, this means that while the number of elements supporting the application can scale very rapidly to meet performance demands, the complexity underlying the application scales in equal proportion. This adds to the uncertainties relating to application topologies that increase the difficulties associated with application support and particularly with root-cause analysis.

Further, many companies actively using IaaS for day-to-day production purposes rely on this elastic infrastructure to scale on demand at the cluster level versus the server level. This means that instead of scaling one server at a time, they may well be deploying 50 or 100 servers at a time. Using "prescale clustering," IT specialists calculate the number of **virtual servers** necessary to support a given number of users. In extreme scaling situations, such as those that retailers encounter on Black Friday, cloud-based servers can then be provisioned as blocks or clusters supporting an additional 1000 to 3000 users. While this type of provisioning can obviously solve immediate problems related to overutilization, it again makes the troubleshooting/root-cause analysis process far more difficult— when problems occur, there are simply far more places to look.

Similarly, in environments such as those characteristic of dynamic scaling or VMware VMotions, servers are provisioned and workloads moved in real time. If a given user is experiencing an application-related problem, it can be extremely difficult to determine which server or class of servers is the culprit.

INTEGRATION

Integration presents unique challenges related to connections between diverse systems. Often, the systems are so different that they have virtually no common features. **Hybrid cloud** is a contemporary example of a complex integration challenge that quickly becomes a concern for nearly every company leveraging **public cloud** to deliver a production service.

Integration points create additional support challenges because each integration point is also a potential point of failure. At the same time, for many application types integration technologies have become the critical glue supporting end-to-end execution.

Web service, ESB, and API connections are all widely used for application and/or data integration. Mobile and containerized applications in particular have ushered in the era of the API, with API-based connectivity being the primary interaction method for container-based microservices. APIs have also become the preferred way of integrating mobile applications with the back-end applications supporting them.

The support problem lies in the fact that monitoring the quality of the end **user experience** relies on the ability of the support tools to monitor the transaction, versus simply the underlying infrastructure. Since this cannot be done at the silo level, this means that monitoring integrations—in context to transaction execution—has become a critical element of both APM and UEM solutions.

APM FOR COMPLEX APPLICATIONS, IN A NUTSHELL

Many IT organizations lack the tools-based visibility, access, and control necessary to manage these complex applications. Without these three capabilities, root-cause analysis and problem resolution become trial-and-error efforts versus relatively simple repair processes.

If visibility, access, and/or control is lacking—as is often the case when components are hosted in the public cloud or by third parties—troubleshooting becomes correspondingly difficult. Keeping in mind the fact that application execution takes a horizontal path *across* hardware and software silos, the following capabilities (which most IT organizations still lack) are prerequisites for automating application management:

- The ability to keep track of the path of each transaction, even in cloud-based or load-balanced environments. This is particularly critical for transactions in which a single error in availability or timing can result in enormous monetary losses, such as with banking transactions.
- The ability to track changing topologies across dynamically changing cloud scale-ups, vMotions, and **software-defined networking.**
- The ability to track topologies across both internal systems, which can be instrumented, and external systems run by cloud providers, partners, or suppliers, which cannot be instrumented.

APM solutions deliver varying levels of insight into application performance from a variety of perspectives, depending on the product. And almost none of the solutions available today cover all three elements detailed in the preceding list. Part of the reason is because, with public cloud IaaS and SaaS in particular, much of the instrumentation necessary for end-to-end transaction visibility is, as yet, not available. APM solutions will become increasingly robust as the data sources underlying execution become increasingly instrumented.

A few specific examples of common instrumentation gaps include:

- Visibility into API performance
- Visibility into dynamic, cluster-based scaling
- Instrumentation of **SaaS/PaaS** performance that is accessible to APM solutions via API
- Visibility into dynamic definitions and changes within **software-defined networks (SDNs)**

That being said, there are multiple high-quality UEM products in today's marketplace supporting one of the most important types of visibility: the ability to see application performance from the end user perspective. In addition, industry-leading APM solutions support comprehensive, machine-generated topologies, sophisticated root-cause analysis across complex applications, and visibility to **API gateways** and similar integration solutions. These topologies deliver the end-to-end views of application and transaction flows that are essential to both APM and UEM.

Developing this end-to-end perspective is not simply a matter of monitoring infrastructure. Often, silo tools indicate that all databases and servers are performing well—yet users still report performance problems. APM tools fill in the gaps—the linkages and relationships between silos—that infrastructure-focused tools miss. By detecting consolidated performance information from the user viewpoint and by relating this information to performance of the underlying silo technology, APM tools support IT organizations in the day-to-day tasks of performance and availability management, root-cause analysis, and problem resolution.

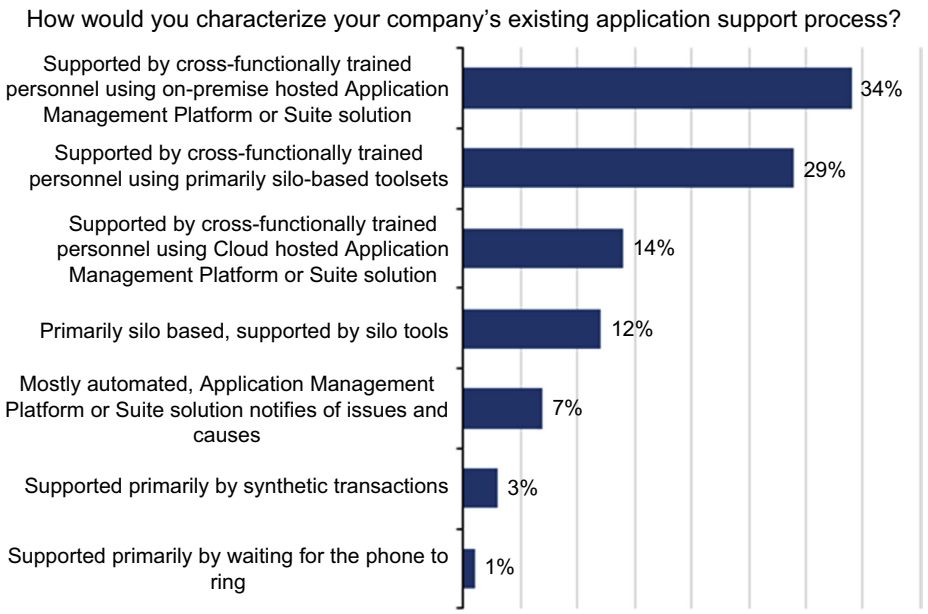

FIGURE 9.4

While most companies support applications with cross-functional teams, fewer than 60% are currently using APM-specific tooling.

In terms of tools, as Fig. 9.4 reveals, fewer than 10% of companies have "mostly automated" application support processes. Nearly 50% are using a combination of "cross-functional teams" and application management platforms or suites. However, that means more than 40% are still trying to manage applications with silo tools and IT personnel who may or may not be application conversant, despite the growing complexity of the applications, transactions, and services running in the average data center.

There are a number of reasons for this. Traditionally, APM platforms are expensive to purchase and deploy. Application-focused skills have been lacking, and the great majority of companies were still holding out hope that they could somehow manage applications at the silo level. Today, the cost and complexity of such tools have diminished as leading enterprise management vendors invested heavily in improvements aimed at making APM tools easier to deploy and use. In addition, multiple SaaS-based APM solutions have now come to market, eliminating the requirement for a massive up-front investment and for ongoing management of the management tools themselves.

"REAL WORLD" APM

As a final example of what the end-to-end big picture actually looks like, Fig. 9.5 shows a topology map for a relatively simple Java-based application. The map was autogenerated by AppDynamics, a leading APM solution that automates monitoring and root-cause analysis for complex application environments.

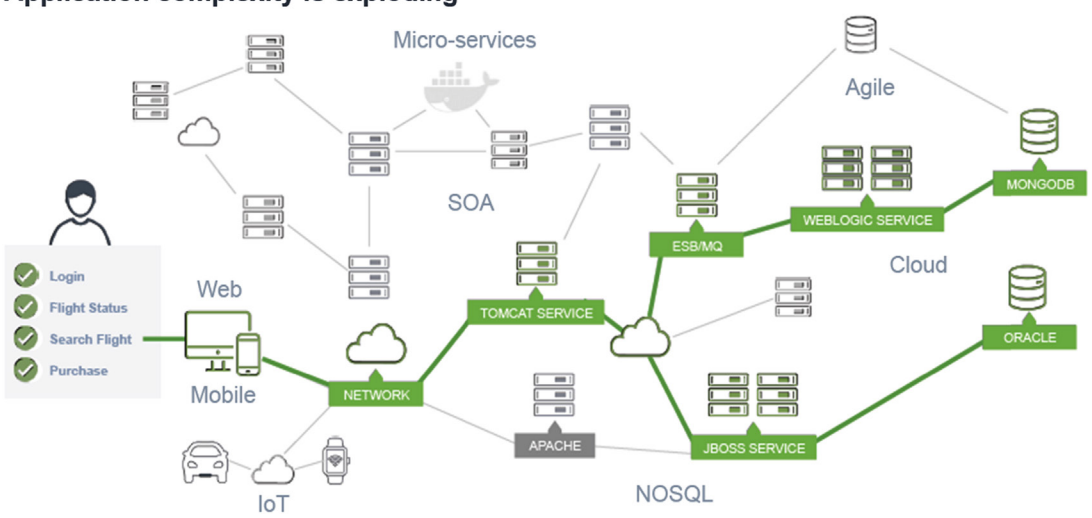

FIGURE 9.5

Automated topology map generated by AppDynamics Platform.

Agents installed on key nodes gather metrics, which are then relayed to a central **analytics** system. In much the same way as a human brain gathers sensory information from a wide variety of sources and makes near instantaneous decisions based on that input, AppDynamics (and similar APM solutions delivered by vendors such as IBM, CA Technologies, BMC, and Dynatrace, etc.) gathers metrics from multiple infrastructure sources and centrally consolidates them into a real-time model of application execution.

This state-of-the-art solution and others like it are now available to IT organizations seeking to automate the extremely complex tasks of topology mapping, troubleshooting, and root-cause analysis.

THE ROLE OF ANALYTICS

Leading-edge application management vendors are well aware that tiered, distributed, and dynamic applications have become too complex for technicians to track and manage manually. IT support teams require automation capable of predicting the impact of change, tracking capacity utilization and trends, and supporting troubleshooting of incidents and problems. Analytics provide the basis for each of these functions.

Analytics have always been a hallmark of the enterprise management space, virtually since the introduction of the first network-focused solutions. These early tools still encountered the requirement to correlate a host of metrics and notifications to a single probable root cause. This early form of "analytics" set the stage for today's analytics processes, which embody extremely high levels of industry knowledge and expertise. Today's tools can assimilate metrics from a wide variety of sources and still

perform the cognitive-esque processes associated with root-cause determination. Analytics processes also assist in **topology modeling**, end-to-end transaction recognition, predictive support activities, and "self-learning" of execution environments.

In addition, APM solutions must not only analyze the metrics from underlying infrastructure supporting the application but also do so in context with the application or transaction running horizontally across the infrastructure. From this perspective, instrumenting the silo layer is just a starting point. The real difficulties lie in tracing transactions *across* the infrastructure, keeping track of dynamic infrastructure and path changes, and doing all of these things in the context of application performance and availability.

The term "Big Operational Data" is used to refer to the sea of metrics being generated by application ecosystems. Data sources include management protocols such as **SNMP** and **WMI**, log files generated by web and Java servers, and proprietary agents installed on devices and supporting systems. In fact, virtually every hardware or software component underlying application execution becomes a source of Big Operational Data input to APM analytics. Due to the wide availability of execution-related metrics, differentiation in the application management space is based primarily on differentiation in the analytics versus differentiation in the metrics themselves.

Leading APM vendors have all developed proprietary algorithms and heuristics aimed at self-learning the environment surrounding the application, topology modeling for the components supporting the application, and **transaction tracing** for tracking transactions as they traverse variable paths across the execution environment. The ultimate purpose of these analytics is to deliver the visibility and control necessary to automate the process of managing applications in dynamic, high-scale environments.

In today's leading edge APM solutions, analytics pull together the complete execution picture, often in near-real time. Application-focused algorithms "understand" how silo performance contributes to end-to-end performance and how the various devices and the software running on them support complex, component-based applications.

SUMMARY

The costs associated with IT have always been viewed as high but were traditionally considered to be a "cost of doing business," similar to utility costs. However, now that technology is the raw material that constitutes a *product*—as it is for social media firms, online services, and other consumer-accessible businesses—the IT functions suddenly become mission critical. At this point, IT budgets start to be viewed as critical investments supporting business growth versus simple overhead.

At the same time, this new emphasis changed the role of IT by adding a new level of responsibility. IT organizations are now key contributors to the business bottom line. This elevation also brings with it a heightened emphasis on accelerating delivery of new services, along with new requirements for improvements in availability, uptime, and performance.

Modern IT services deliver substantial business value. At the same time, they also magnify IT's support burden. Progress is never free, and in the case of SOA and distributed applications, the business reaps the benefit while IT pays the price. In lieu of automation, IT turns to manual support processes to keep pace. The result is clearly illustrated by a discussion between an EMA analyst and a mid-level manager in the healthcare industry. The support team's manager described spending 30 hours on a root-cause analysis conference call while team members worked to troubleshoot and resolve an

application-related problem. Clearly, efforts of this magnitude eat away at IT budgets and are unacceptable to businesses focused on IT services as sources of revenue. High-quality APM solutions can eliminate the need for calls of this nature.

KEY TAKEAWAYS

- The definition of the word "application" is exceedingly broad and defined differently in a wide variety of contexts. Often, the only thing "applications" have in common is the fact that they are created from software code and designed to perform a discrete task or set of tasks.
- Complex, component-based applications comprise the majority of "applications" supporting virtually every enterprise-sized company. At the same time, their diversity, scale, and complexity make them very difficult to support.
- Heterogeneity, complexity (i.e., multiple moving parts), and scale are key support challenges in managing "modern" applications.
- Topology models relating transaction execution to underlying infrastructure can dramatically simplify troubleshooting and root-cause analysis. Lack of such a model means IT support teams are "flying blind" in terms of application support.
- Although supported by multiple infrastructure and software elements, component-based applications cannot be managed at the silo level; managing such applications requires visibility to end-to-end execution *in context to* underlying technical elements.
- Approximately 40% of companies are still trying to manage applications with silo tools; this becomes untenable as complexity of the underlying infrastructure scales out.

Examples of vendors with products in this space:

Apica
AppDynamics
AppFirst
ASG
Appnomic
Aternity
BlazeMeter
BMC
CA Technologies
Dynatrace
Dell Quest
EMC
ExtraHop
HP
IBM
Idera
ManageEngine
Nastel

New Relic
NetScout
Netuitive
Oracle
Push Technology
Riverbed
SmartBear
SOASTA
Splunk
Stackify
Sumo Logic