An Oracle White Paper
April 2010

# Bridging the IT Visibility Gap in Complex Composite Applications

## Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

## INTRODUCTION

As more and more IT shops deploy multitier middleware platforms and frameworks for the delivery of business-critical application services, they are expecting to garner the benefits of the service-oriented architecture (SOA) or composite application development approach: agility, flexibility, productivity, and extensibility. Typically, these benefits are delivered to application developers by the framework vendors, using proprietary or standards-based protocols. This flexibility for users usually results in complex multitier environments, and the responsibility of managing this complexity has been assumed by developers and IT staff.

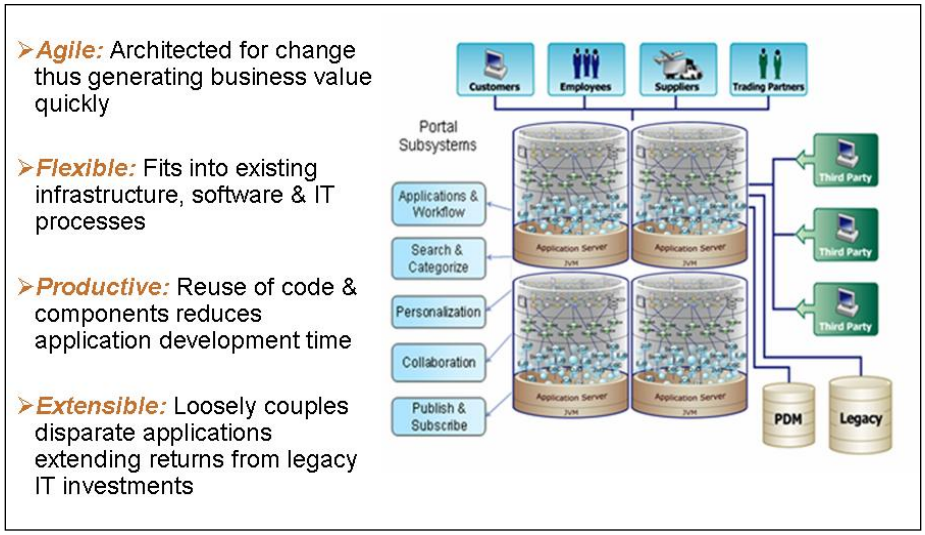


**Figure 1: Expected benefits of a composite application or service-oriented application approach**

Implementing composite applications or SOA is expected to align IT with the business and quickly adjust to change as competitive pressures arise, as illustrated in Figure 1. However, the legacy application management products currently supporting these critical applications are increasingly keeping these organizations from achieving these benefits. With the broad acceptance of Simple Object Access Protocol (SOAP) and Representational State Transfer (REST), composite applications are further extended, especially because they allow the application logic to be further subdivided or delivered by external environments.

Traditional application performance management (APM) products provide deep visibility into the performance of an enterprise application's code components, but these tools fail to correlate the code and code component performance with the performance of the application services delivered by these complex composite applications. For example, an application implemented with a portal framework on top of an application server delivers specific application services to end users, who are represented on the portal pages by tab and page flow combinations.

Although a traditional APM solution generally enables an IT organization to pull metrics on the underlying code supporting an enterprise application, it typically doesn't allow the IT organization to set a performance metric on that particular application service or business function, such as the account balance query, and then provide correlation down to the code components supporting that

service. In *n*-tier composite applications, this correlation from the application services to the code would provide IT organizations the ability to both diagnose and optimize the performance of their applications on a service-by-service basis. This is critical, because one application service on a particular portal page may be performing fine while another application service may be underperforming, yet they are leveraging shared application components.

The composite applications of the early part of this decade leveraged a relatively simple and static architecture, generally a database with an application server and a graphical user interface. Today's composite applications are significantly more complex and dynamic. A typical SOA enterprise application today might leverage a Java Enterprise Edition (Java EE)-based application server with a portal engine, a Service Bus, a Business Process Execution Language (BPEL) engine, and perhaps an alternative integration framework to connect to a legacy back end. Each of these application components is, in its own right, an application, and with a SOA-based application, the components are bound to change on a regular basis. Additionally, the shared-services code components constituting each of these application components (such as entity beans, JavaServer Pages, and portlets) support multiple application services, possibly across multiple applications.

Using traditional APM products to correlate an application service with the underlying shared-services code components enabling that service has become all but impossible. This correlation is critical, however; the very purpose of an enterprise application is to provide business services, and if a problem with the performance or availability of a particular application service supported by a composite application cannot quickly be triaged, the value of the application will soon be overcome by the cost of outages, poor performance, and general maintenance chaos associated with supporting the application.

Additionally, IT organizations need a dependable means of performing impact analysis. For example, before a planned change is made, IT operations must be able to identify all application processes, application components, systems, and so on, that will be affected by the change. This adds significant value only with the necessary context to bridge that gap (the "IT visibility gap") between application services and components.

Today's composite applications require a new management approach that dynamically correlates application services with the shared code components supporting those services. Although a standalone APM approach will suffice for relatively simple Java EE applications, APM alone has become obsolete for managing SOA and other complex composite applications.

This white paper describes why a Composite Application Management approach is critical to providing enterprises the hierarchical visibility necessary to effectively manage today's composite applications. It explains why this approach makes it possible to automate the instrumentation and dashboard creation, thus simplifying and commoditizing APM tasks that were once extremely time-consuming. Additionally, it illustrates that with today's increasingly complex, multitier composite application architectures, enterprises must stop investing in legacy APM approaches and start investing in Composite Application Management solutions that encompass capabilities traditionally associated with APM solutions while providing those capabilities in an automated and scalable fashion.

## A HISTORY OF APPLICATION PERFORMANCE MANAGEMENT

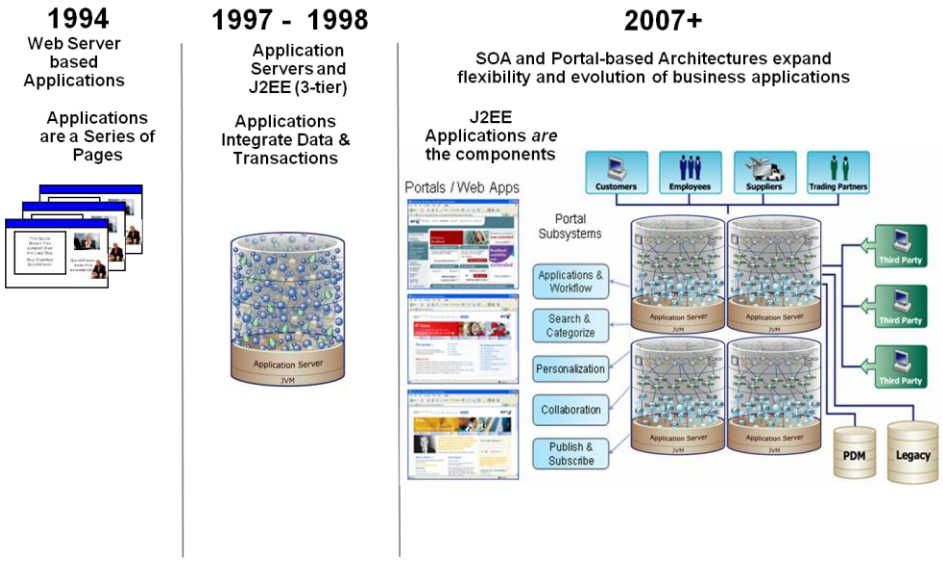


**Figure 2: Evolution of J2EE applications**

During the 1990s, most enterprises developed their mission-critical applications by using object-oriented programming languages such as C++. Managing these applications did not generally present much of a challenge; because the IT development teams managing them typically built the entire applications, they could easily develop a tool that had visibility across whole applications.

In the late 1990s, the application server market took off, with BEA WebLogic, IBM WebSphere and other application server platforms gaining significant ground. These application server platforms provided much of the plumbing required to build a scalable enterprise application, leaving IT shops to invest in the business logic and differentiating capability that provided the most value.

Although the emergence of the application server market substantially accelerated application development schedules, it also made managing production applications a little more challenging. For the first time, a big portion of application code was not written by the IT organization developing the application, necessitating a management translation layer from the application server vendor's code to the application services provided by the application. A new breed of companies emerged in approximately the year 2000 to address this need, providing manual byte-code instrumentation packages that enabled IT architects, operations teams, and developers to drill down into the underlying Java virtual machines (JVMs) within the application servers containing the business logic and pull out performance data on the code (see Figure 2). These vendor tools typically instrument everything within an application infrastructure and then require professional consultants from the companies to reconfigure the instrumentation and build custom dashboards, bringing the monitoring overhead down to a reasonable level and correlating the code-based metrics with the services provided by the monitored application. As long as the application architectures were not terribly complex, this solution worked fairly well, providing enterprises much more visibility into their application performance than they had had without these tools.

## A NEW APPLICATION LANDSCAPE

In the early part of this decade, many CIOs began to express concern that they could not measure the return on their IT investments. Of course, businesses overspent on software and technology in the late 1990s, and CIOs began to demand that their development teams and software vendors demonstrate financial return. In 2002 Mercury Interactive Corporation (now part of Hewlett-Packard Company, or HP) recognized this opportunity and launched a new product suite based on a concept it called business technology optimization (BTO). Essentially, BTO meant providing CIOs and other high-level IT executives with financial visibility into the performance of their application investments to ensure that IT was aligning with the business.

BTO tracked key performance indicators (KPIs), service-level agreements (SLAs), and other high-level metrics that spanned application investments but typically did not look into the applications. IBM Corporation; Computer Associates, Inc.; and HP soon followed with their own BTO-like offerings. In the meantime, BMC Software led the charge with business service management (BSM), which was built to help IT organizations understand the business relationships between their application services and the IT value these services delivered. BSM was based on automating IT service management (ITSM) processes, the heart of the Information Technology Infrastructure Library (ITIL). However, neither BTO nor BSM provides the contextual relationship between an application's services and the code components enabling those services.

Figure 3 shows the challenges posed by the IT visibility gap.

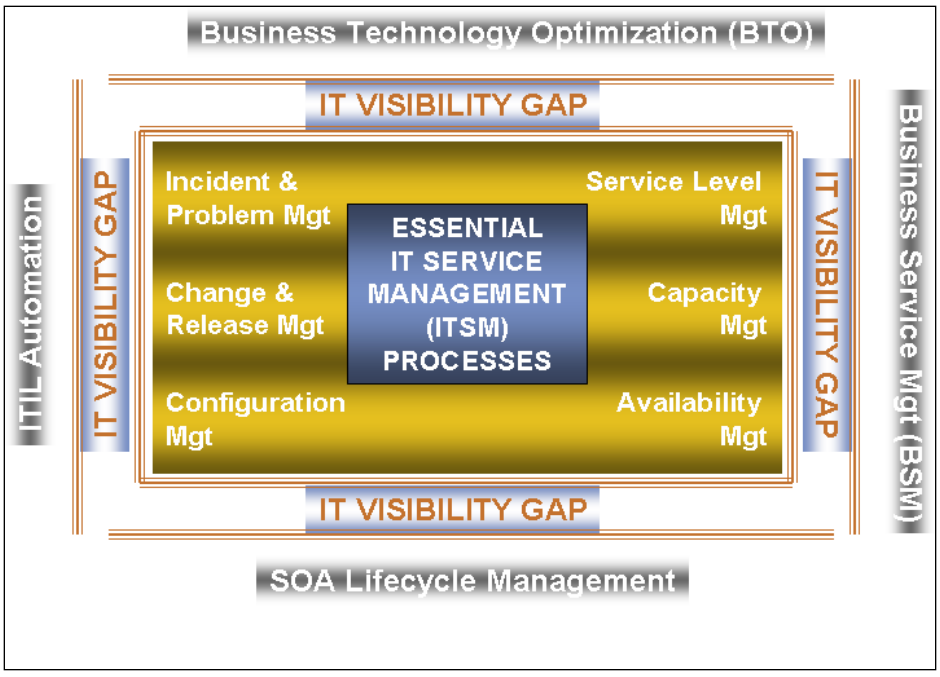


**Figure 3: The IT visibility gap must be overcome to achieve BTO, BSM, and other essential IT service management processes.**

Without this context, IT organizations deploying complex composite applications cannot effectively align IT with the business, leveraging the tools and processes in which they have so heavily invested. The IT visibility gap has emerged as a result of the move toward shared application components,

which, by design, hide the relationships between components in multiple layers of abstraction. With all the benefits expected from SOA, new application management challenges have arisen.

In the early 2000s, with the emergence of the application server market and the deployment of first-generation composite applications, all an enterprise needed in order to effectively manage its composite applications and application investments was a good APM solution and perhaps a BTO solution. By 2005, however, the leading middleware platform vendors, such as IBM, BEA Systems, and Oracle, started to introduce specialized engines on top of their application server platforms, which provided significantly more capability to enterprises looking to build rich applications. Portal frameworks such as IBM WebSphere Portal and BEA WebLogic Portal dealt with the front-end functions of human-to-machine Web interfaces. Integration frameworks dealt with integration into back-end systems such as legacy mainframe applications. Even more recently, SOA frameworks such as ESBs and BPEL engines enabled the delivery of Web services and the rapid introduction of new services into an application without necessitating a wholesale change to the application architecture.

In theory, the net effect of all of these middleware enhancements enables an IT team to quickly build a sophisticated enterprise application and constantly roll out new services for the application without having to rewrite key application components. Although building a rich and highly sophisticated application has become much easier with these *n*-tier middleware components, managing the application has become much more complex.

Specifically, as the middleware environments have continued to become more sophisticated, they have created a management gap between the management of the resources (APM) and the management of the business services (BTO). Today metrics have become a commodity. Most application servers spit out Java Management Extensions (JMX) data, making code-level performance visibility easy. With complex composite applications, however, the importance of understanding the specific, unique, and dynamic associations between the application services and the shared code components that support those services has superseded the importance of having metrics on either the code or the application services.

## THE NEED FOR COMPOSITE APPLICATION MANAGEMENT

The greatest challenge with managing these complex composite applications is gaining the necessary visibility to enable IT to perform its essential functions: incident and problem management, change and release management, capacity management, availability management, service-level management, and configuration management.

Figure 4 illustrates how three major factors—layers of abstraction, specialized expertise, and constant change—contribute to the IT visibility gap.

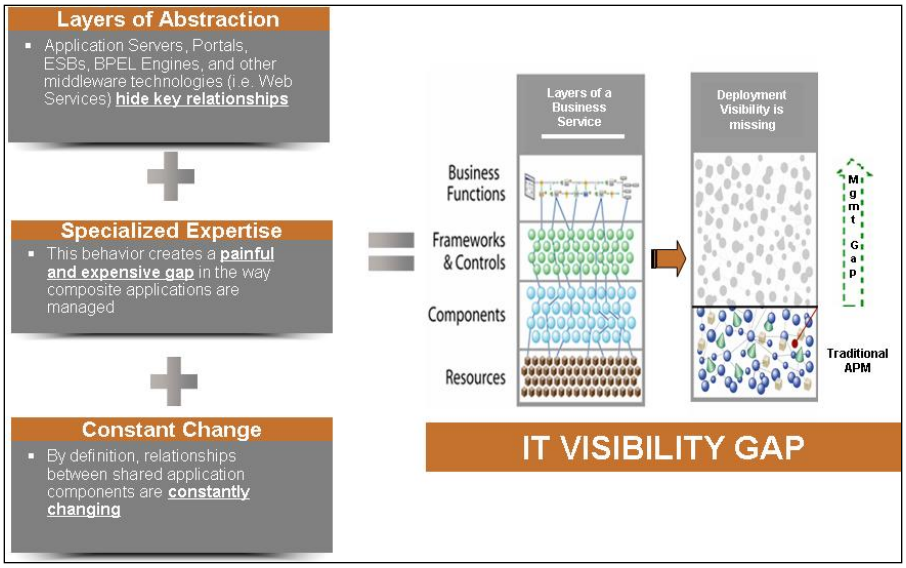


**Figure 4: The problem—complex composite applications without visibility**

As mentioned earlier, the various middleware technologies and off-the-shelf code components organizations have invested in are hiding the key relationships between business functions. This is most apparent once the application development team hands off the enterprise application to the staging and production operations teams. The tools for building these complex applications are not built for the monitoring, testing, and ongoing management requirements of IT operations.

Most important, traditional APM and system management tools have no way to correlate application-provided services with the underlying code components supporting those services, because the tools focus on breaking open the JVM and collecting and displaying metrics at the code level without the business context. Unfortunately, IT must then manually link the relationships together and build dashboards to illustrate the relationships between the code-based metrics and the application services. Of course, all of this requires detailed knowledge of all the different code components, middleware technologies, and relationships between applications. Worse, the manually linked relationships quickly become obsolete as the applications change.

The pace of application change today is measured in days or weeks, whereas the typical time for analyzing a performance problem by using a traditional APM solution is measured in weeks or months. This makes for a formula that simply doesn't work. It creates a very painful and expensive gap in the way composite applications are managed. In many cases, the gains achieved in the development of today's composite applications are often tossed away during the deployment and management phases of projects.

Figure 5 illustrates a shopping cart application in which three business functions are used by the business service: search, quote, and order. The search function can be leveraged by an employee planning to purchase polo shirts for loyal customers, as illustrated by the red line. The quote function may be triggered by one of the company's distributors looking to generate a quote for an upcoming indirect order via a reseller channel, depicted by the blue line. Note that the path taken leverages one or

more shared application components. Finally, the order function, shown by the black line, represents a customer order directly from the shopping cart application.

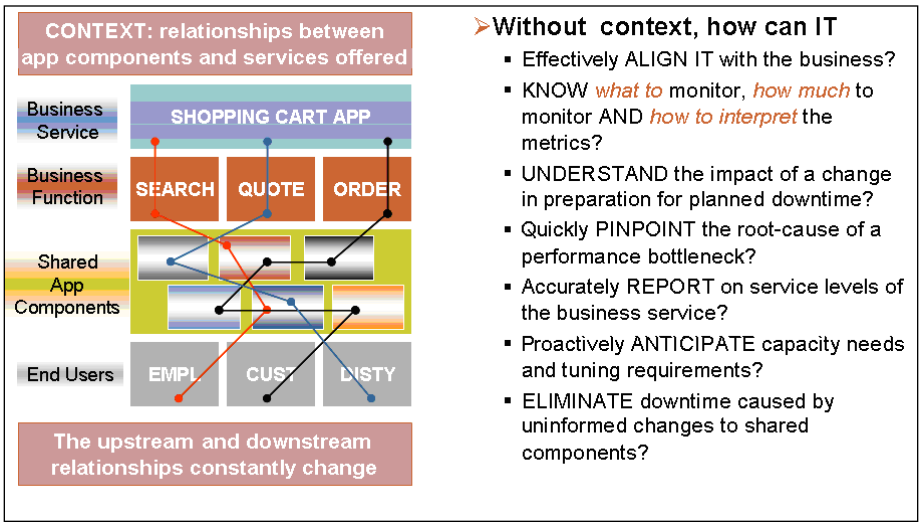


**Figure 5: The importance of context**

Figure 5 shows that, depending on the context of the request, various business functions leverage one or more shared application components—taking different paths—when delivering services to end users.

Without understanding the relationships between the application components and the services being offered ("context"), how can IT effectively align its actions with the priorities of the business? How does it know what to monitor, how much to monitor, and how to interpret the metrics? How can IT effectively prepare for a downtime window when it does not know what other services may be affected by the change? What about capacity management or measurement of service levels? How about pinpointing the root cause of a performance bottleneck?

Referring to Figure 5, imagine a situation in which one of the company's distributors or resellers cannot receive quotes from the shopping cart application but other business functions being served by the same shared application components work fine. Determining the root cause of the problem involves determining the specific code paths and associations between the services on that reseller's Web site and the portal, integration, workflow, and J2EE code components.

To make it even more challenging, complex composite applications are constantly changing, along with the upstream and downstream relationships between components.

## COMPOSITE APPLICATION MANAGEMENT IN ORACLE ENTERPRISE MANAGER

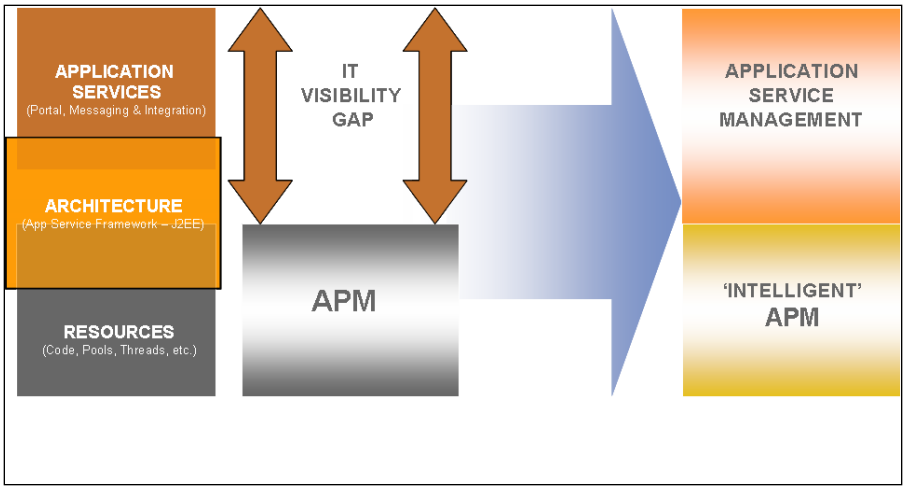


**Figure 6: Bridging the IT visibility gap**

Oracle can bridge the IT visibility gap by delivering Oracle Enterprise Manager, which correlates the services provided by the applications with the underlying code components supporting those services. In other words, Composite Application Management provides the context required to effectively manage the performance of complex composite applications.

Leveraging Composite Application Management, Oracle Enterprise Manager can intelligently determine the appropriate monitoring points, deploy the required agents, and dynamically display role-based dashboards with relevant metrics within the context of the application services being delivered. Because of the inability to capture the context, no one else in the systems management space has been able to attack this management challenge from the top down.

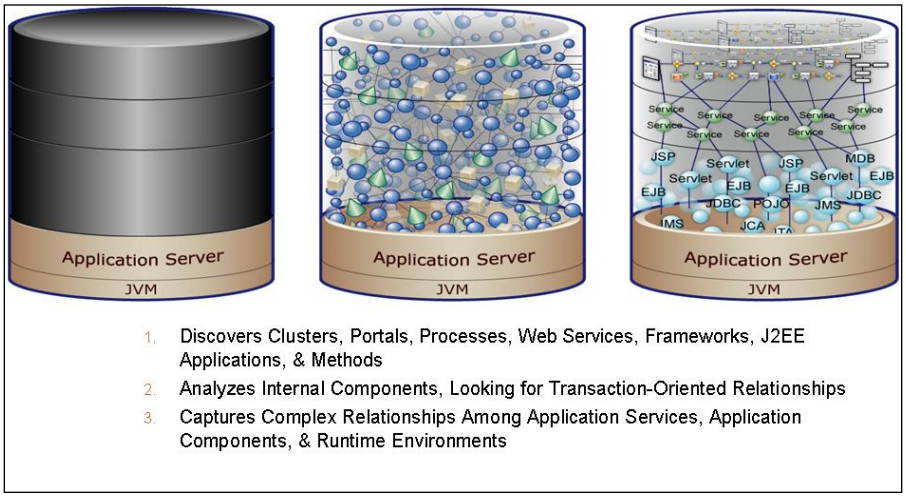


**Figure 7: The Oracle Enterprise Manager modeling process**

Oracle is able to accomplish this by first building a topological model of the application architecture—mapping the business functions down to the code components that deliver them (see Figure 7). More specifically, Oracle Enterprise Manager, once connected to the administration server of an Oracle

WebLogic domain, a WebSphere cell, or another application server environment, discovers all the JVMs and associated application servers under management. All the critical files are copied to the Oracle Enterprise Manager which performs both static and dynamic analysis on the application. The static analysis identifies all possible code paths available. The dynamic analysis identifies the actual code paths being used at runtime and the application-to-application relationships and picks up on-the-fly entity and structural changes that occur.

By creating a model of a composite application and understanding how business transactions flow through the different layers, Oracle Enterprise Manager automates the key steps of composite application management (setup, analysis, change management, and service-level reporting). Leveraging this model, Oracle Enterprise Manager uses sophisticated algorithms to determine the optimal instrumentation points across the enterprise application, from the application services supporting the enterprise application all the way down to the code components. Additionally, Oracle Enterprise Manager automatically builds management dashboards, enabling both drill-up and drill-down capabilities (see Figure 8).

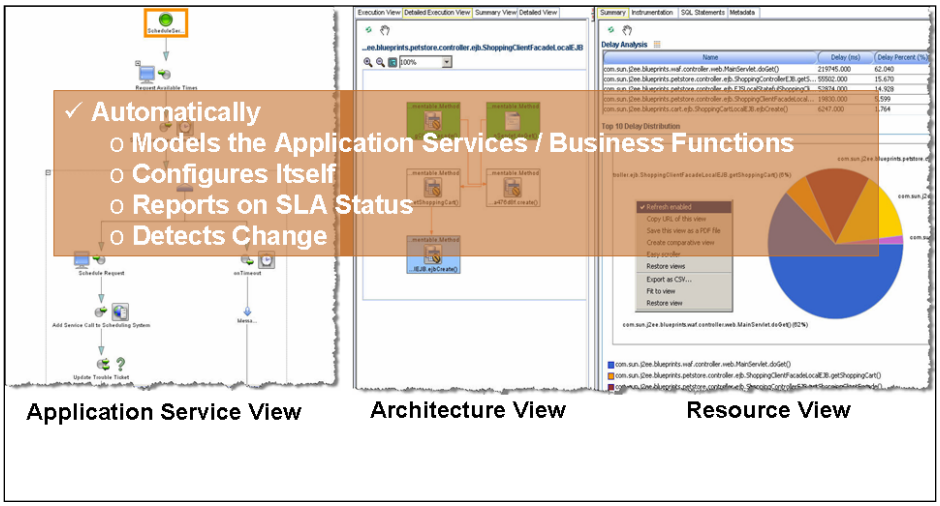


**Figure 8: Oracle Enterprise Manager provides a complete Composite Application Management solution**

Because Oracle Enterprise Manager uses a model-based approach, people doing preproduction and production operations can immediately determine the root cause of a performance problem on a particular application service and then determine through Oracle Enterprise Manager's correlation capabilities what other application services might be affected. In addition, Oracle Enterprise Manager enables IT operations teams to set performance metrics at the application service level, at the code level, or anywhere in between, and the status of these metrics is automatically correlated across the Oracle Enterprise Manager dashboards.

In summary, Oracle Enterprise Manager automatically models the application services, configures itself, reports SLA status, and detects changes. It benefits the entire IT organization throughout the lifecycle of enterprise applications—development, quality assurance, and production.

## CONCLUSION

The combination of multiple layers of abstraction, the need for specialized expertise, and constant change create an IT visibility gap, inherent in today's complex composite applications, that can be addressed only with Oracle Enterprise Manager, which bridges the gap. Without this context, it is impossible to effectively diagnose application service problems in enterprise applications that leverage multiple layers of middleware. An unmanageable application, no matter its sophistication or richness of services, provides little value to an organization. Further, an IT operations team must understand an application's logic to effectively conduct the impact analysis required to stay ahead of potential application problems. Only a Composite Application Management approach, coupled with fully automated APM, enables IT organizations to realize the value of their composite application investments.

Oracle Enterprise Manager leverages Composite Application Management to provide the following automated capabilities:

- Automatic context mapping and change detection
- Automatic monitoring point determination and deployment
- Automatic role-based dashboard creation

By leveraging Composite Application Management, Oracle Enterprise Manager enables enterprises to maximize application service levels, reduce the mean time to resolution, and attain the expected return on investment from their application investments.

An investment in Oracle Enterprise Manager not only manages the many layers of abstraction facing enterprise IT operations today but will also adapt to one of the most disruptive layers of abstraction to ever hit IT: server virtualization. For more on this topic and how to effectively manage tomorrow's virtualized multi-server clustered composite applications, look to the Oracle Enterprise Manager approach to Composite Application Management.

Bridging the IT Visibility Gap in Complex Composite Applications
April 2010

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment