APM WHITEPAPER

Why APM?

Because Performance Management is Not Optional

Compuware
THE TECHNOLOGY PERFORMANCE COMPANY

Compuware APM

Why APM?

Because Performance Management is Not Optional

BY ROB VOLLUM

The evolution of technology and related applications has fundamentally changed the world over the past three decades. It's hard to imagine a single person disagreeing with that assertion. During this time, applications have become increasingly sophisticated, growing from single-purpose tools into interconnected business systems reaching far beyond the data center. The audience of users has grown from internal technical and business specialists into the population at large. Today, people (young and old) and businesses (large and small) use online applications as an integrated part of daily life. This is how we communicate, collaborate, educate, buy, sell, shop, research, schedule events, work, entertain ourselves, and just about every other thing imaginable. Technology and its applications have become infrastructure, just like electricity and running water. Considered in this way, it becomes clear that organizations must manage and protect what have become key assets – their applications. Application Performance Management, or APM, is no longer optional. For organizations with broad customer and partner interactions, failure to have a solid APM strategy is simply irresponsible.

As applications have evolved, so has APM. For example, the name itself has grown from Application Performance ***Monitoring*** to Application Performance ***Management***. Monitoring performance implies a reactive approach, based primarily on alerts raised by tools designed to manage siloed software or hardware components, and primarily serving the Operations team. These operational alerts are the basis for the all-hands-on-deck war room "blamestorm" sessions, with technology representatives scrambling to isolate and solve the problem at hand (with a bit of proving, "It's not me." thrown in). Managing performance, on the other hand, means taking a collaborative lifecycle-based approach in which development, testing, and production/operations work together to not only solve problems when they occur, but to proactively avoid them in the first place. The focus is on the application as a whole, along with its supporting processes. It requires having the same rich information available for analysis in each of those teams.

When answering the question, "Why APM?" there are two good approaches to take. One is to answer in the context of increasingly complex technological requirements. In other words, answer the question as it pertains to APM supporting the sophisticated application development and delivery process. The other is in the context of the application owner who is making this investment -- a "What's in it for me?" answer. Both of these approaches have value, and we'll explore both.

Application Complexity Drives the Need for APM

To understand the requirements for APM, and how they have changed over time, we need only look at the shape of applications over the years. As applications go, so goes APM.

Centralized Client-Server Applications

Early applications were centralized, typically on large mainframe or mini computers hosted in the environment in which they were used. Graphical interfaces were simple, starting as "green screen" terminals and eventually moving up to PCs in client-server fashion. Application architecture was easy to understand, characterized for example as such:

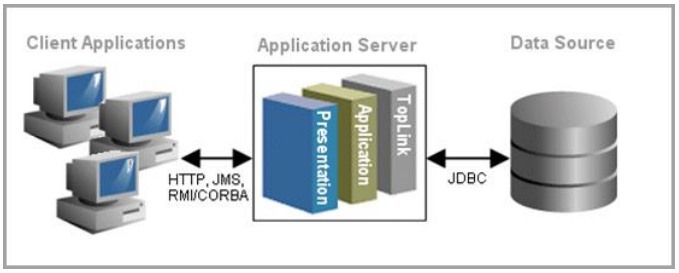
Client-Server / Mainframe Architecture

APM, at this stage of application evolution, was essentially non-existent. Host or server monitors would provide reports on server utilization for things such as memory, CPU, disk, I/O bandwidth, MIPS, and so forth. Applications, on the other hand, were triaged when they failed (or were slow). This was typically handled through some manner of logging, as well as system debuggers if the problem was severe.

Three-Tier Applications

Bracketing the turn of the 21st century, we saw the next major evolutionary step in enterprise applications: the three-tier model. In this model, client applications would run on PCs (fat client and browser-based). The client applications (handling presentation)

connected to an all-encompassing middle-tier application server (handling business logic), which would itself interact with a backend database tier (for required business data).



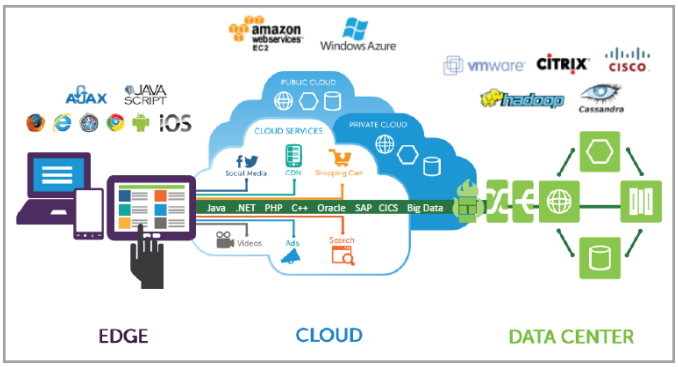
Three-Tier Architecture

Architecturally, this presented a simple, easy to understand application topology. In this model, the application server tier – generally J2EE (now called JEE) or Microsoft .NET – was organized as a small cluster of JVM or CLR instances hosted on a small number of servers.

Given the inherent complexity of the new J2EE and .NET managed platforms, combined with the added complexity of event-driven object oriented programming, it quickly became clear that a new approach was needed to keep these different applications performing efficiently. It was at this time (roughly circa 1998) that the first generation of APM tools came to market. These tools used a variety of new approaches to provide visibility into these "managed code" environments. Challenges included not only providing visibility into the application code, but also providing visibility into the frameworks of the platform (database access, business object management and persistence, memory management, and so on) as well as into the operation of the platform itself (via JMX/PMI metrics in J2EE for example). From the outside, synthetic transactions served to model representative response time of key transactions. Internally, thread stack trace sampling and the first uses of byte-code instrumentation came into use for unprecedented visibility into the running application. Additionally, network protocol analysis helped visualize the application footprint across the monitored infrastructure.

SOA-Based Applications

The mid-2000s brought us to the current state of the industry in application architecture when SOA, or Service Oriented Architecture, gained traction. SOA-based applications are granular and highly distributed, packaging individual functions (such as login authentication) as independent APIs, often accessed as Web Services. Unlike the three-tier model of the late 1990s, transactions in SOA-based applications can easily and routinely cover 10-15 heterogeneous tiers to complete their action. Complicating things further, some of those tiers may belong to partner organizations, accessed via the cloud. Consider, for example, an airline's website offering rate quotes for rental cars, directly from the rental car companies.



SOA-Based Architecture

As mentioned previously, application architecture has a direct impact on the requirements for APM. This major change from three-tier to SOA-based applications had a correlated effect in the APM market. The most immediate (and painful) effect was with the first-generation APM tools. Those tools were designed to manage a fixed small number of nodes on a fixed small number of physical servers, where transactions generally executed entirely within a single (or small number of) monitored instances (JVMs/CLRs). Virtualization and "the cloud" compounded this complexity, and many first-generation APM tools (the pioneers) were not able to redesign their products to work in the "new normal" where servers and nodes can come and go, based on load, with hundreds or thousands of monitored instances and servers, in local data centers or in the cloud.

The second effect was the emergence of the second-generation of APM tools, designed with this "new normal" in mind. With past experience as a guide, leaders in this generation have raised the bar and have completed the shift from *Monitoring* to *Management*. Lifecycle approaches provide a feedback loop from development through production. Low-to-zero configuration and dynamic expansion and contraction for "elastic" applications support agile development methodologies and a DevOps mentality. A focus on Business Transactions allows analysts to assess business impact of application slowdowns. Gathering of historical performance data allows for IT analytics, helping us move from reactive to proactive to predictive in our management of key applications.

It Doesn't Stop There

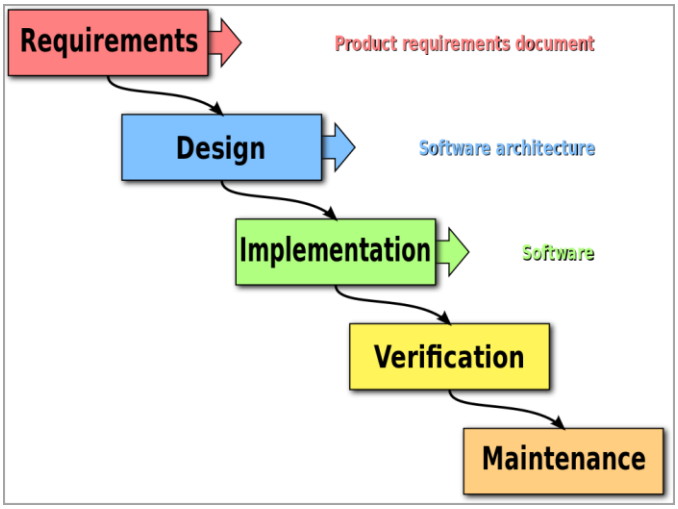
The progression from three-tier to SOA-based applications, along with the ever-growing impact of the Internet and cloud-based computing, forced a fundamental change in what is considered acceptable APM. It represented an "all bets are off" call for change. We have reached the point where humans can't easily visualize or comprehend the complex transaction flows of large, distributed, event-driven systems without specialized tools. Coupled with that, user expectations are higher than ever. For example, people often expect better performance from smartphone apps than from desktop applications. Feature-wise, users look for turnaround in weeks, not months. Without clear lifecycle visibility into their existing applications, companies cannot release new features fast enough.

What will be the next drivers in the APM evolution?

- "Big Data". The volume and type of data being gathered is going beyond relational database standards. NoSQL databases add flexibility to analytics
- Ongoing sophisticated use of the cloud for elastic computing resources. APM solutions need to integrate seamlessly. Expanded Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) offerings are driving more companies to the cloud
- DevOps and the Agile SDLC: It's normal now for organizations to make 10 production releases a day, versus one release per quarter. APM solutions need to be designed in from the beginning to provide deep details with minimal overhead, across the product lifecycle
- IT analytics. The data gathered by APM solutions is a wealth of information to be mined, to move to the predictive level of APM

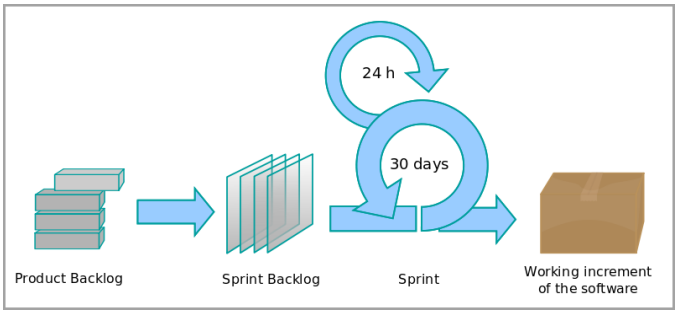
Speed of Innovation Drives the Need for APM

As application architecture has become more complex, it should come as no surprise that the related Software Development Life Cycle" (SDLC) has become more complex as well. In the days of architecturally simpler, (monolithic, cast in stone), systems, developers were well-served with the waterfall SLDC methodology. Using the waterfall approach, the process began at the beginning and moved ever forward, until the software product was released to production and entered the maintenance phase. A key point to this approach is that there is no going back to previous stages once they are marked complete.



The Waterfall SDLC Methodology

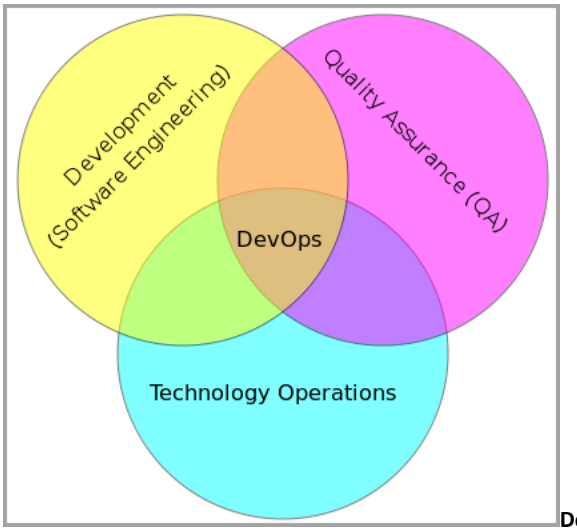
As applications progressed towards SOA, it was clear that the waterfall method could not work. Systems were too big and changed too fast to be able to rely on such a tedious, inflexible, and documentation-heavy SDLC model. Increasingly, product teams adopted an "Agile Development" approach. There are several popular Agile SDLC methodologies, each differing in its specific approach. However, they all share the core principles as captured in the "Agile Manifesto", a set of guidelines written by a small group of Agile advocates in 2001. Some of these popular methodologies are Extreme Programming (XP), Lean Development, Feature-Driven Development (FDD), and (pictured below) Scrum.



Scrum, an Agile SDLC Methodology

The key to any Agile SDLC approach is the early-and-often feedback given based on the many iterations involved. Rather than being cast in stone, as with the waterfall method, participants are free to learn from experience and factor that back into the product design. The product is tested frequently, minimizing the risk of future failure.

Finally, in conjunction with advancing SDLC methodologies that support complex application architectures, a new partnership has evolved between the development, QA/test, and production operations teams. This partnership supports what is known as the DevOps approach to product release management. DevOps stresses communication, collaboration, and integration among these three teams, with the end goal of rapidly deploying software products and services. In efficient DevOps organizations, it's not uncommon to see ten or more deployments daily.



DevOps

So what exactly does all of this have to do with APM? Everything! With accelerated development cycles, there is no dedicated "performance phase" during which we can test each component. With releases moving from one every few months, to several releases a day, engineers need confidence that they will not

inadvertently introduce performance flaws that may go undiscovered for several releases. And finally, a requirement for DevOps success is that members of all three teams must have access to all relevant data in its native format (i.e. not screenshots or attempts at re-creation) serving as a common language for performance management.

It is these complex and fast-moving circumstances that that make APM a requirement in today's enterprise applications. In order to support the ongoing demand for "more, better, faster", increasingly capable APM solutions are required, with the additional requirement that they keep up with ever-evolving processes and procedures.

Benefits of an APM Solution

We've now seen that in order to keep up with increasing application complexity, coupled with the need to support increasingly compressed development and release cycles, APM is required. APM necessarily becomes part of the platform. That gives us a technological answer to "Why APM?" Let's take the next step and consider the "What's in it for me?" answer to "Why APM?" The short answer is "plenty". Here's a top 10 list of benefits, ranging from technical to business-facing.

Benefit: Deep visibility into the application during normal as well as underperforming operations. The always-on nature of second-generation APM tools means that critical problems aren't missed, requiring production support teams to try to recreate the issue, or in the worst case, just wait for it to happen again. Being able to see into the application hotspots removes guesswork. Additionally, data gathered over time allows for trending and pattern identification, providing insight into questionable areas before they become problems.

Benefit: Elimination of the war room "blamestorming" sessions -- a standard scenario in the non-APM world is a pager going off for 30 people, all of whom convene in a conference room (war room), or dial in via a conference bridge. Often vendors will participate as well. With no visibility into the application, the challenge is on for live troubleshooting largely to prove "It's not me." An APM solution provides the ability to isolate the problem and assemble just the team or teams that are truly required. Attention is paid to solving the problem instead of proving who didn't do it.

Benefit: Adds predictability into the troubleshooting process. Conventional wisdom is that the goal is to reduce MTTR, or "mean time to repair". This is a standard KPI in production environments. This is a useful metric, however, if you are focused primarily on MTTR, then you are still in reactive mode: how long does it take to fix this thing that's broken? With mature APM solutions in place, organizations can focus on a more useful metric: MTBF, or "mean time between failures". If MTBF is high, then MTTR becomes less of an issue. To paraphrase an old saying "If it ain't broke, you don't have to fix it."

Benefit: Reduced operational costs. This benefit shows in many ways. For example, a standard defense against poorly tuned applications is to throw more hardware at them -- more CPUs, more memory, more servers, more VMs. All of these cost money. Overtime or consulting services spent on triaging poorly performing applications cost money. Penalties and breach-of-SLA fees cost money. Any efforts made to fix problems caused by the lack of an APM solution will effectively rob the budget.

Benefit: Business continuity. Given management by a solid APM solution, application owners can offer uptime SLAs and be confident that applications will remain working efficiently as designed. This provides business continuity, since the application is "open for business". Clearly, when an application is down unexpectedly, business is interrupted. In the brick-and-mortar world, common sense and personal experience tell us that if people never know for sure when a business will be open, they will find another business. So it goes for online businesses as well. An application that is technically alive, but not able to transact business is no better. Brick-and-mortar retailers are keenly aware that out-of-stocks (inability to buy what the customer came in to buy) is the number one cause of customer dissatisfaction. In the online world, applications that are not able to complete transactions in a timely fashion push customers to alternate sites that are just a mouse click away.

Benefit: Ability to establish believable and actionable SLAs with key stakeholders. Trending data allows us to know what's "normal", and build the trust that we can maintain that level.

Benefit: Higher productivity due to reduced performance-related wait times. Numerous studies show that the human attention span is short when it comes to online interactions. Response times of under a second are considered instantaneous. One to two seconds is acceptable. Beyond that, people start to lose their focus. Productivity drops due to a lack of concentration (at best) or due to distraction by outside influences (at worst). By the time they get back to it, they start with the "Now where was I?" process. Applications that operate efficiently, avoid this problem by letting users complete the task at hand and then move on.

Benefit: Every significant online site or enterprise application has a purpose, with "desired results". Those results could be increased sales, higher revenue, more downloads, etc. Applications managed by a solid APM solution will keep users' attention and interest, and lead to more of these desired results

Benefit: Improved end-user experience and greater customer satisfaction. When it's boiled down, APM success is directly measured by the end-user experience. A successful APM solution will translate directly into improved end-user satisfaction.

Benefit: Ability to quantify the real time business impact of application execution. Leading APM solutions are able to tie into contextual application data as well as application performance data, tracking the key business transactions of the application. That means, in addition to knowing the performance of your *doPurchase* transaction, you can also know what products have been bought, or

the city of the purchaser. This leads to some real time business intelligence based on live application data. Performance problems in the application will be immediately reflected in the live dashboards, reflecting over-time behavior.

Risks of Not Having an APM Solution

In some sense, risks and benefits are just the opposite side of the same coin. Since risks can lead to costly situations, it bears discussing them directly. As with the benefits, we'll arrange the risks from technical to business-facing.

Risk: With no APM solution in place, every performance issue is a fire drill. With no visibility into the application, there is no way to assess the magnitude of the problem, nor pinpoint its origin. This is hard enough in a single- or three-tier application. Imagine the difficulty in a SOA-based application with users on the Internet and 70 tiers in multiple datacenters and in the cloud! With no APM solution, there is no reasonable way to estimate how long problems will take to resolve.

Risk: High loss of key employee productivity due to hours on triage in the war room. When important applications go down, or have serious problems, it's an emergency. When there is no visibility into the application with an APM solution, the only option is to assemble the best and the brightest, from all functional areas of the application. The time they spend locked in the war room is time they are not spending doing other things, such as making applications more robust or making money for the company. Instead, the company is losing money with a down application (with no idea why it's down), and the best technical people are wasting time trying to diagnose the problem(s).

Risk: It's impossible to understand the flow throughout a large-scale distributed SOA-based application. Troubleshooting is at best silo-based, with no application context to assist in the process. Log files offer the best insight, but finding the right ones to begin with, and working to correlate them across the various other silos, is a tedious, daunting task.

Risk: The inability to innovate! Production problems and "application down" issues take priority over everything. When these systems are down, the company is down. Nothing is more important than that. Without an APM solution in place, we have no warning when these situations will take place, and when they do, we have no idea when they will be resolved. Lack of visibility into the live systems means they are fragile and easily broken. And since the best and the brightest are collected in the war room patching them, they aren't working on your market-beating new initiatives. This risk is hard to measure in real dollars. It's only measureable over time as your competitors innovate their way right to the top.

Risk: The inability to set meaningful SLAs. In today's collaborative world, partnerships are key. Think of travel rollup sites that offer integrated airline, hotel, and car deals. Those sites live and die by the promises (SLAs) made to them by their partners. With no APM solution in place, there is no meaningful way to track what is reasonable or expected performance over time or to be confident in understanding how long it will take to fix a problem once it is reported.

Risk: No data to analyze application performance trends. Similar in nature to the inability to set SLAs, this broader risk means that we have no visibility into the ups and downs of our application performance. What are the peak times, during the day, week, month, season? Do we need more capacity or less? Do we need to spin up a presence in the cloud for better elasticity? The ability to trend performance data – IT analytics – is a big element of APM, with heavy emphasis on the "M" for management. This will support the action from reactive to proactive, to predictive.

Risk: Loss. When an organization ignores the need for a comprehensive APM solution, it risks losing many things, including:

- Customers and related revenue. Competition is high online, and customers vote with their clicking finger. While customers will remain loyal through an occasional inconvenience, inconsistency will drive away the most loyal of fans
- Business partners. When you look bad, they look bad. And your partners will not be willing to share in the blame of your poorly performing website.
- Respect and credibility in the industry. Serious organizations today realize that having a modern APM solution in place is not negotiable. Not taking that seriously is a red flag for customers, partners, and anyone doing business with the company. Most recently, our own federal government stumbled here with the rollout of www.healthcare.gov. The results speak for themselves.
- Stock price/valuation. It's reasonable to think that organizations that don't realize the importance of infrastructure may have problems with other fundamentals too.
- Competitive advantage. It's hard to innovate when you're always fighting fires. It doesn't take long for focused competitors to climb past these encumbered organizations.

Risk: Lawsuits. In today's hyper-connected world, online applications have become infrastructure, like electricity and running water. Large organizations have a responsibility to their shareholders to have a strategy for keeping their online assets available and open for business, especially when they guide consumers there as a primary choice. To do otherwise is irresponsible.

Summary

The question "Why APM?" can be answered from two perspectives: the technical (to support innovation and manage application complexity) and the business (for the value derived versus the risks avoided). From both perspectives, we converge on the single conclusion that APM must be a key component of application architecture today.

To those making these critical decisions – you are not alone. Consider the opinions of leading analysts on the subject:

"Complexity and visibility of the infrastructure are causing a transformation in APM technology. In addition, customers are demanding lightweight, smarter, automated solutions that provide rapid return on investment, analytics and support for business demands."
— Jonah Kowall, Research Director at Gartner

"APM is a cornerstone investment in both processes and technology that I&O [Infrastructure and Operations] leaders must make to better manage the applications that run their business and serve their customers"
— Jean-Pierre Garbani, Principal Analyst Vice President, Forrester

In closing: in under a decade, applications have shifted from an internal focus to corporate assets that are directly impacting the business more than ever. APM has shifted from being just another tool in the IT toolkit for reactive troubleshooting, to a strategic C-level initiative for proactively assuring superior application availability and performance for end users around the world. Revenue, loyalty, and brand image have never been more dependent on how well our applications perform. That, in a nutshell, answers the question "Why APM?"