Buying APM in the enterprise

A survival guide from a veteran performance geek

APPDYNAMICS

Buying APM in the enterprise

A survival guide from a veteran performance geek

About the author: Jim Hirschauer led several enterprise-scale application performance management (APM) projects at large financial services institutions as a Monitoring Architect before joining AppDynamics as a Technology Evangelist in 2011. Jim was brought into the Investment Banking division at a large FS organization to help reduce the number of incidents impacting end users. Within a few months, Jim was able to get the organization headed down the path of APM maturity, and within a couple years the organization was proactively seeking out and fixing performance bottlenecks as well as dynamically adapting to change in workload demands. During the time Jim was a Monitoring Architect, he saw a significant improvement in response time for multiple applications and reduced the number of incidents that impacted customers by 90%.

Introduction

Many organizations have a stockpile of monitoring tools they've either built themselves or bought, but most of them are practically unused. There are many reasons that software becomes shelfware – maybe the person who brought it in or built it has left, or perhaps it didn't meet expectations, or it could simply be out-of-date – but it's painful to see it happen. It's not an efficient way to use a product, especially if the company invested anywhere from thousands to millions of dollars licensing and maintaining it.

The goal of any IT organization is to implement tools that continue to drive value from multiple employees and across teams, even years after the software was purchased. This Survival Guide aims to help make that vision a reality by teaching you how to properly identify, vet, and select software that will have a real business impact. In this Survival Guide I'll share my personal experience as a buyer of application performance management (APM) software, including the lessons learned from both my triumphs and my failures (that's right, it's okay to fail as long as you fail fast, learn from your mistakes, and never repeat them).

Part one

APM maturity as you've probably never seen it before

Part one: APM maturity as you've probably never seen it before

Maturity models usually fail because they're too theoretical. Vendors shove maturity models at their customer in a last-ditch attempt to improve adoption and retention rates, but the customer is simply too busy solving problems to care. That's why I've put together my own maturity model for APM, one based in real-world experience fighting fires and using APM tools rather than theories about how APM should be used. In this section I'll present my new maturity model and provide some example questions that an APM buyer or user might ask in each stage.

The best indicator of where you are in the maturity model is determined by the kinds of questions and statements that that come up in your organization. For example, when my child asks where babies come from, I know just about where he belongs in the Life Maturity Model—and any other child asking that question is probably in about the same stage regardless of age. To make it easier to identify where you and your organization lie, I've organized my maturity model by the types of questions you might be asking at each stage in the process.

Hirsch's APM maturity model

Level 0 – WTF just happened?

- We just got a bunch of phone calls that the Website/ Application is slow. Really?
- CPU, memory, disk, and network all look great.
  Why is it still so slow?
- You start making phone calls or start/join a conference call asking
  -Did you change anything?
  -Do you see anything in the log files?
  -Are we having network problems?
  -Can someone get the DBA on the line?
  -It has to be the database!
  -It just started working again. Did anyone change anything?
  -Is it fixed?
- 3AM phone call from help desk…It's broken again. Damn!
- How does a business transaction relate to IT infrastructure?

Level 1 – Ouch, too much information!

- Our new monitoring tool sure does provide a lot of data. Look at all these charts to spend hours digging through.
- It took a long time to set up all of those alert thresholds but I bet it will be worth it.
- Why so many alerts? Did everything break at the same time?
- Is anything really wrong? I don't know, go test the site/app to find out.
- It worked great in dev/test/qa. What's different about prod?
- We profiled our code in dev and it is still slowing down in production. Why???
- It's still slow for our customers? It looks fine from the office.
- Our APM tool is okay for testing but we wouldn't dare use it in production.
- Does anyone know what dependencies exist between our applications?
- I heard about something called DevOps. Any idea what it is?

Part one: APM maturity as you've probably never seen it before

Level 2 – Whew, that's getting better!

- We're still getting a lot of alerts but now we know if apps are slow or broken.
- We don't set alert thresholds very often; our tooling alerts us automatically when important metrics deviate from their baselines.
- Looks like some of the functions in our app are always slow. Let's focus on optimizing the ones that are used the most or are the most important.
- We built a dashboard for our app to show when it gets slow or breaks.
- We can see everything going on in test and prod and know what's different between environments.
- We know if any of our end users are impacted because we monitor every business transaction.
- Yep, the problem is on line 45 of the DoSomething method.
- We automatically deploy monitoring with our apps. It's part of our build/release process.
- Our applications and their dependencies automatically get mapped by our tools. No need to guess what will break if we make a change.
- Wouldn't it be cool if we could automatically react to that spike in workload so our site won't slow down or crash?
- I wonder if the business felt any impact from that problem?

Level 3 – APM rockstar

- We built a business AND technology dashboard so that everyone could see if there was any impact at any given time.
- All of our monitoring tools are integrated and provide a holistic view of the health of each component as well as the entire application.
- Whenever there is an application slowdown (or when we predict there will be a slowdown) due to spikes in user activity, our tooling automatically adapts and spins up new instances until the spike ends.
- When any of our application nodes are not working properly, our tooling automatically removes the bad node and replaces it with a new functional node.
- The data derived from our APM toolset is used by many different functional groups within the organization spanning both technology and business.

From looking at these questions and statements, you can probably identify which level you and your organization belong to. Even more importantly, you may have an idea about how to advance along the maturity path by looking at what you'll have accomplished by the next step in the model. Obviously, utilizing APM software is an essential part of attaining higher levels of APM maturity, but good processes and well-trained people are also critical components of success.

In the next section we'll take a look at how to get started down the path of deploying APM in the enterprise so that you can advance your monitoring maturity and realize significant value in your business.

Part two

Getting started with APM

Part two: Getting started with APM

If you've looked at the maturity model presented in the previous section, you'll have some idea of where you and your organization lie. Working on improving processes and training your staff can help you to advance along the maturity model to a certain extent, but in order to see real progress you'll need to get insight into your application's performance with APM software. In this section I'll present my tried-and-true method for getting an APM tool into your environment, based on some key lessons I've learned from my own experience dealing with various software vendors.

1. Find the pain

Nobody will agree to spend money on a tool unless there is some problem actually hurting your business (e.g. lost revenue, productivity impact, customer satisfaction). If you want to justify a purchase, find a tangible problem and document it. This will preferably be an issue with a business/mission critical application such as your eCommerce platform, online trading application, payment gateway, risk calculation, settlement system, etc...

Find some application or service that is impacting your business in a meaningful way due to poor performance and/or downtime and document the following:

- Number of issues and severity level
- Mean Time To Repair (MTTR – usually the average amount of time from first impact to problem resolution)
- Quantifiable measure of impact on business (e.g. dollars lost per minute, potential customers lost, trades lost per minute).
- Average number of employees involved in troubleshooting each issue
- Root cause of each incident

You will use this data in your evaluation document and your business justification down the road.

2. Take stock of what you have

Many IT organizations have dozens of tools in their possession that are rarely, if ever, used. If you haven't already done so, you need to take inventory of what software you already own and document your findings. You will use this information for years to come as long as you keep it up to date. Your inventory should address:

- What tools exist and what category should they belong to? (e.g. Database Monitoring, Network Monitoring, OS Monitoring, Desktop Monitoring)
- How many licenses do we have and are they current?
- What are they good at?
- What are they not good at?
- What would be classified as an APM tool?
- If I already have an APM tool why is it not being used properly?

Put labels on your existing tools and understand what they do. This will help to identify where your weaknesses really are, and if you have any tools in your possession that aren't being fully utilized already.

Part two: Getting started with APM

3. Uncover your blind spots

Now that you have the overall landscape of your monitoring ecosystem laid out, you need to see if there are any gaping holes. One way to do this is to compare your existing toolset against a model of what an application performance management solution should include, like Gartner's 5 Dimensions of APM*. Gartner's model includes the five following criteria for a "complete" APM solution:

1. End User Experience Monitoring: Measuring the response time of your application all the way to the end user. It's not good enough to just understand how fast your application runs within the confines of the data center(s).
2. Application Topology Mapping: Automatic detection and display of all components involved in the delivery of your application. You need to know what application components are in use at any given time, but especially when there is an issue impacting your users.
3. Business Transaction Profiling: Detecting and measuring the response time of all application component activity initiated by a single user request. This is not the same as measuring the response time of a web page!
4. Deep Application Diagnostics: Detecting and measuring the run time code execution within your application containers. If your current or prospective solution does not load into the application container you will NOT have this important capability.
5. Analytics: Intelligence applied to data that provides you with actionable information. This is not the same as reporting, and analytics can (and should) be a key differentiator between competing solutions.

Gartner's model should give you an idea of what you're looking for in an APM solution. Most software doesn't incorporate all five aspects of APM on its own, so many organizations use a combination of different tools to get the complete visibility they need. Take a look at the tools listed in your inventory to find out where the holes are in your APM strategy.

4. Narrowing the field

So now you've identified the pain–that giant, pesky performance issue that keeps coming back to haunt your app–and you've figured out what tools you have in your inventory and where their blind spots are. Now it's time to begin looking at new solutions, but where do you begin? A Google search for "application performance management" comes up with 37 different companies and 97,000 results. Where do you even start looking?

First, you'll need to narrow down the candidates to 2 or 3. This is traditionally done with feature comparisons, phone calls with analysts, references from people you know, crystal balls, voodoo rituals, and some form of animal sacrifice. In my experience, however, the best way to narrow down the field is the hard one: creating a spreadsheet. To do this, just open your favorite spreadsheet tool, then fill in the columns with your potential vendors and the rows with your requirements (keep them pretty generic for now). Here are a few common APM requirements to get you started:

- Automatic detection, naming, and monitoring of Business Transactions
- Automatic discovery and deep instrumentation of application code
- End user experience monitoring
- Analytics based alerting
- Automatic discovery and display of my application topology
- Support for my application technologies
- Support for my application architecture (e.g. cloud, monolithic, distributed)
- Open-ness of vendor (Did they skirt around my questions? Did they talk in circles?)

*Similar models can be found for other areas of IT, but the focus of this Survival Guide is APM, so that's what we'll discuss here.

Part two: Getting started with APM

Sample requirement chart

	Vendor 1	Vendor 2	Vendor 3
Automatic BT detection	Yes	No	No
Automatic code instrumentation	Yes	No	No
Supports Java?	Yes	Yes	No
Open-ness of vendor	Very open – pricing on website	Did not give a quote	Got a quote after talking to sales
Good alerting?	Yes – intelligent	OK – must configure	Poor – Universal thresholds
Works in cloud?	Works well in cloud/hybrid	Does not work in cloud/hybrid	Works well in cloud/hybrid
Support for SaaS?	Yes	No	Yes
Production overhead	Good	Too high	Good
Easy	Easy	Difficult	Easy

Once you have this matrix built it should be easy to narrow your choices down to 2 or 3 vendors and you're ready to begin with the Proof-of-Concept (POC). But first, some advice about keeping your vendors honest…

5. Showing, not telling: Get the demo

You may notice that the requirements in the previous section were pretty generic. That made it easier to quickly narrow down the vendors without spending too much time researching them. However, now that you've identified your 2 or 3 top vendors, it's time to get a little more specific with your requirements. When you engage with a sales rep you'll need to be very explicit and detailed with your questions as they apply to your environment. If you get a partial answer from a vendor or a response that is not quite what you are looking for, make sure you dig deeper right away to figure out if they are blowing smoke or not. If they keep talking in circles you can be pretty sure they can't really do what you are asking.

The reason for this, as you probably know, is that APM vendors can get creative in their marketing. This isn't new, or even unique to APM, but finding the truth about a product can be especially difficult in this industry because it's so difficult to get your hands on the real thing. That's why I recommend that you always make them show you what you're looking for in a live demo.

I'm not saying all vendors are evil liars. Sometimes simple miscommunication can lead to disappointment if the promised feature exists but doesn't meet your expectations. The vendor needs to earn your trust, and eliminate any possible confusion about what the product can and can't do, by showing you the product in action.

Part two: Getting started with APM

6. The proof-of-concept

Now that you've seen the product in action and know that they weren't lying about meeting your requirements, it's time to start the Proof-of-Concept (POC). The POC demands its own set of much more detailed requirements. This is the time to really look under the covers at a product and see what it can do versus what you have already been told. Again, using your favorite spreadsheet tool create a new sheet for each tool in the POC. I prefer to use a weighting and grading scale to help differentiate between tools:

Weighting: Some requirements will be more important to you than others. I use a Low, Medium, and High scale with associated values of 1, 2, and 3. This is your multiplier for the grade you give each requirement.

Grading: Some tools have better support for a requirement than others. I use a grading scale of None, Poor, Adequate, and Excellent with associated values of 0, 1, 2, and 3.

I also like to break up the POC requirements into categories. This helps keep me organized, ensures I don't miss testing something on my list, and helps me write the analysis document at the end of the POC. Some examples of categories and requirements are shown below.

- Authentication and Authorization
  - -Supports Microsoft AD
  - -Granular Role Based Permission System
- User Interface
  - -Web based user interface compatible with Internet Explorer 8+
  - -Displays application topology without administrator or user configuration
- Deep Application Diagnostics
  - -Automatically discovers and instruments custom code
  - -Automatically traces complete call stack when performance is abnormally poor
  - -Automated intelligence to ensure instrumentation does not use excessive overhead

Your list should be much longer and more detailed. This requirements list is the basis of the POC and all follow-up documentation, so make sure it is thorough and not slanted towards a particular vendor. Think of it as your Christmas list when you were a kid: don't be afraid to ask for things that might seem impossible but that could be really useful.

Another key lesson learned coming your way… Don't let the vendor control the POC. You define the environment (Dev, Test, and even Prod if it can be done safely), you do the installation, and you do the configuration. In short, you do just about everything related to the POC. You are the one who has to use the tool after you buy it, so be sure to personally drive the POC initiative.

And lastly, make sure to provide the same playing field for each vendor so that your results are really comparable. Having a successful POC using a team of 5 vendor engineers versus another successful POC using just 1 engineer is comparing apples and oranges. Keep the playing field level to make sure you're giving each vendor an equal chance.

Part two: Getting started with APM

7. Pick your champion

After you wrap up all of your POCs, you should have enough data to pick a winner. Hopefully you built your spreadsheets so that they automatically add up the numbers related to all of the requirements. You will have a statistical winner based purely upon your spreadsheet data, and that usually aligns with the overall feeling you have after the POC is complete. Sometimes, however, you may have a vendor/product that is statistically the winner but your gut is telling you to choose a different vendor. In this case you need to figure out exactly why you feel this way. You cannot justify vendor selection based upon a gut instinct alone. Maybe one vendor was just a royal pain to deal with, or maybe the solution worked well but was agonizingly painful to deploy. The key here is being able to express why your instinct is pushing you a certain direction and **put it in the evaluation document (we'll talk about this more in a minute)**. Something similar to the following statement can be used as justification of your position:

"Even though Vendor X Product statistically scored highest in the evaluation there is one overriding factor that prohibits us from selecting them as the overall winner. Deployment and configuration of Vendor X Product is difficult and time-consuming. Based upon the observed deployment and configuration time of 2 weeks for 1 application during the POC, it would take approximately 19 years to configure monitoring for the 500 applications that are in scope. Deploying Vendor X Product does not make sense in our environment."

The information you have been building throughout this entire process should be used to create an overall evaluation document. This document should have the following information at a minimum:

- Description of problem
- Description of proposed solution
- Vendors/Products evaluated
- Evaluation Criteria (Requirements)
- Evaluation Results
- Recommendation
- Next steps

It can also be helpful to create a short (3–10 pages) presentation to accompany the evaluation document that you can use to brief management on your findings. This presentation should contain only the most important facts since all the details are in the full evaluation document.

Another important document to create is the business justification document. The business justification cuts out all the technical details related to the product you want to purchase and gets right down to the economics of the matter. I am not going to dive deep into how to write a business justification; however, to help you get started, you should make sure you get a ROI (Return On Investment) calculator from each vendor that participates in your POC. Vendors want to help you buy their solution and will provide a wealth of information to help you build your business justification, so just ask them for help if you need it. The trick is to make sure that everything in the business justification is based in fact and relevant to your business.

Lastly, it is really helpful to have broad support for your initiative. Seek out people across your organization that will support your recommendation of the product or who will validate the problem that you are trying to solve. If you have full support of an Application Owner whose business is being directly impacted, this greatly increases your chances of success.

Requirement	Importance L=1 M=2 H=3	Grade None=0 Poor=1 Adequate=2 Excellent=3	Score Importance x Grade
Authentication and authorization			
Supports Microsoft AD	L	3	3
Granular Role Based Permission System	M	2	4
User interface			
Web based user interface compatible with Internet Explorer 8+	H	0	0
Displays application topology without administrator or user configuration	H	2	6
Deep application diagnostics			
Automatically discovers and instruments custom code	H	3	9
Automated traces complete call stack when performance is abnormally poor	H	1	3
Automated Intelligence to ensure instrumentation does not use excessive overhead	H	2	6
Total			**31**

Conclusion

If you're still reading this, congratulations! You've completed my Survival Guide to Buying APM in the Enterprise. Hopefully you've learned something from my experience as an APM buyer and owner and you can now feel confident in your path to becoming an APM rockstar. In this Buyer's Guide, we've learned a few strategies to selecting an APM tool that will be successful for your organization long after you've gone. We've covered:

1. Defining success using an APM maturity model
2. Identifying and documenting the problem to justify the purchase
3. Finding your blind spots and taking an inventory of your current tools
4. Defining your APM requirements and seeing how the vendors stack up
5. Getting proof during a demo
6. Playing fair during a POC
7. Choosing your champion

Hopefully this guide has helped you in your journey to choose an APM tool that will help your organization drive real business value. Now get out there and start rocking!

APPDYNAMICS

www.appdynamics.com