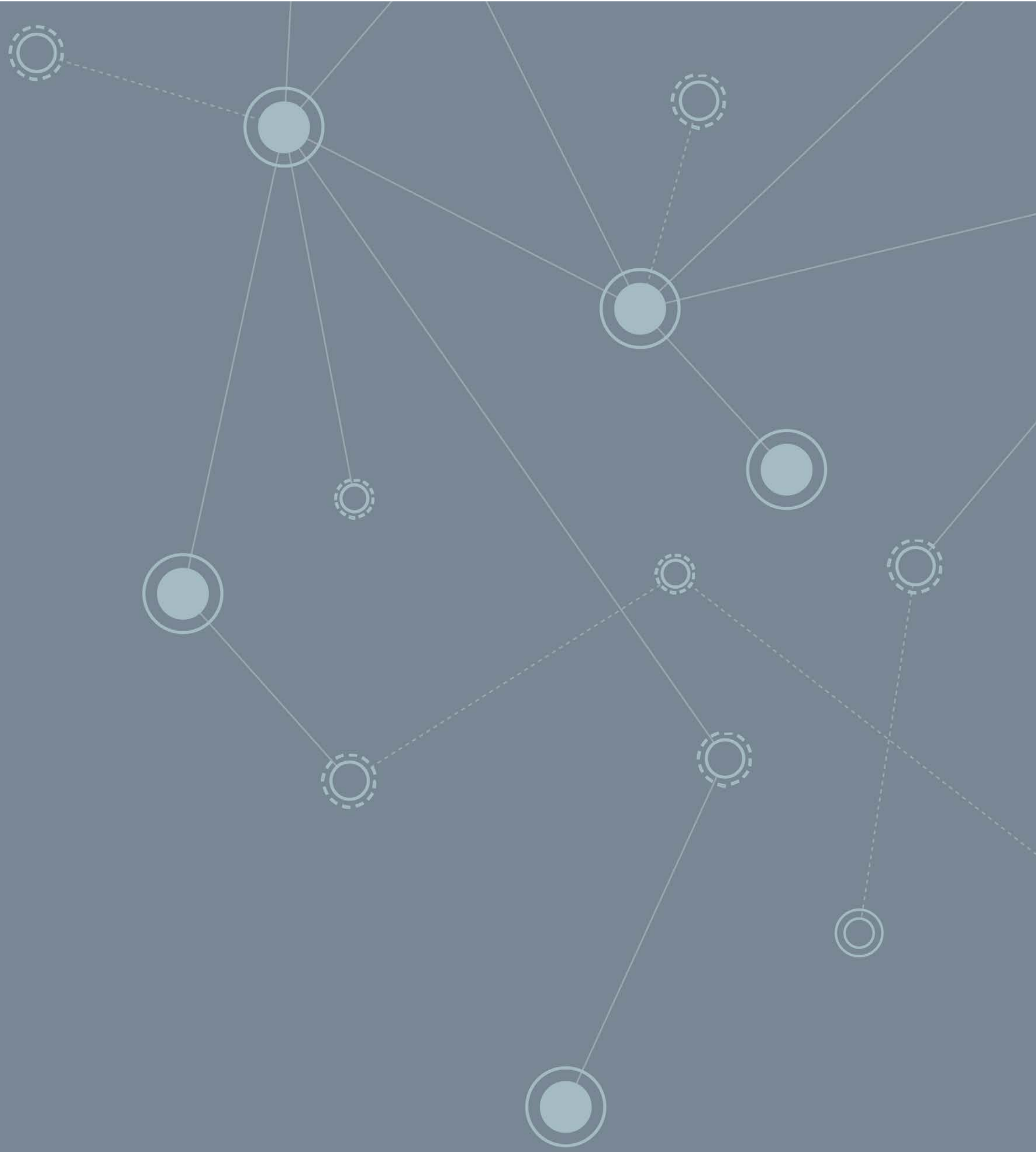
APPDYNAMICS

Keep CALM and Embrace Devops



Introduction

Culture
Automation
Lean
Measurement
Sharing

For the last couple of years, every conference I attended and nearly every call I had at Forrester had a discussion on DevOps. There is a good reason for this, because from my perspective, DevOps promises that the software-powered business can deliver quality app-fueled business services rapidly, to delight customers.

I would also say that the term DevOps is not encompassing enough as it's not just about development and operations. It's also about the business, because in today's digital business, every function is accountable for software strategy success. Take a look here at this blog by my colleague Anand Akela for our view on BizDevOps.

The CALMS Framework For DevOps

In most discussions I have had with ops professionals, one question came up again and again in regards to DevOps, and this was "Is there a framework for DevOps adoption?" Now there are good reasons for this question and one is that many enterprise ops folk are aware of frameworks such as ITIL. But rather than a framework, ITIL morphed itself into a whole, daunting library of books. This is something, which I hope never happens with DevOps!

But there is one framework that has emerged which was originally coined by Jez Humble (@jezhumble), a pioneer in DevOps. Jez devised the CALMS framework, which makes perfect sense to me:

C – Culture
A – Automation
L – Lean
M – Measurement
S – Sharing

Culture

Let's Start With Culture

Culture – One of the most used words in business today. *"It's all about the culture… ",Culture is how organizations 'do things;…", "Culture is about rituals….",* etc., etc. Well, that clears up what culture means?!?!? The reality is that culture is such a fluffy term that it means something different to different circumstances, people, businesses, industries, countries, etc.

To define culture and what it means for DevOps you have to understand the starting point, or the challenges that your enterprise's operating model currently faces in regards to software strategy and what the end, new 'DevOps' operating model looks like. Here are a couple of examples:

1. **Shift from a fear of failure, to a fail-fast, fail-forward approach.** DevOps is about speed. If your operating model today promotes zero failure and employees are scared of failing then you will stifle the ability to innovate, to promote new ways of doing things, to move fast. Failure can be good so long as we learn and improve. This is all part of innovation, which is central to DevOps.

2. **Shift from a tech obsession, to a customer obsession.** In today's digital world, it can be so easy to get caught up with tech buzzwords such as mobile, wearables and cloud. But the rules of business have not changed. Deliver what your customers want, delight them, and hopefully they will help to promote your brand. This means that every employee has to think about the external customer, their needs, their wants in order to guide strategic decisions. It's about moving from an inside-out to an outside-in operating model.

3. **Shift from organizational silos to a collaborative model.** Let's face it, the desire for collaboration across different business functions has always been a goal. But collaboration is never as good as it can be. I could write a whole book about why, but largely this is because business = people = different agendas = politics = failure to collaborate. To move fast, deliver quality software rapidly, and then it's essential we get collaboration right. This is not just about collaboration between Dev and Ops, but an operating model that promotes collaboration across business functions (e.g. digital teams, marketing), development and operations.

4. **Shift from big data confusion to real-time information driven insight.** In the fast moving world of DevOps information and insights in context will be your business lifeline. This means that having application data such as engagement, technical and business data (revenue etc.) changed quickly into information that can be consumed by different business audiences is essential to making fact-based strategic decisions. So we have to move away from the current confusion around big data and analytics and shift to an operating model that makes an analytics solution that focuses on applications, a core part of making strategic decisions in regards to software strategy.

Why APM and Analytics is Key to DevOps Culture

Having worked with many enterprises and APM solution vendors in my time at Forrester, I believe that a great APM solution can support all four of my points above. We are at a turning point in the APM market as APM is no longer just about incident management and response but is about making sense of application data, turning it into insight to support business decisions. A great APM solution today has analytics baked in and it has to be simple to use i.e. simple to turn data into information, simple to display information to different audiences. At AppDynamics, we uphold three core principles for our application intelligence platform:

See - Our platform is able to safeguard and optimize application performance from the end-user (customer or employee) through to the infrastructure workload and database/data store backend. This means that we can detect potential issues before they impact the customer. This supports a fail-fast, fail-forward operating model in a customer-obsessed business.

Act - Our platform includes automation features meaning that the business can respond quickly to potential problems that could impact the customer. So for example, if an application server is being maxed out, we make it easy to automatically spin up another server before a performance or outage issue. On top of this our war room feature makes it simple for the business, development and operations to collaborate looking at the same information in real-time, to resolve issues quickly.

Know - Our integrated application analytics makes it easy to display information in context of the audience. We collect all application data and make it simple to for technical or non-technical audiences to change this data into information so that strategic insights can be made.

Automation

I want to turn my attention to the next letter in the CALMS model – the letter A for Automation. If you boil down DevOps to its basics then it's all about releasing new applications or features at speed while maintaining quality. In order to do this, automation solutions are vitally important but.....

I have problems with Automation

For me the mantra of achieving speed via automation tools is nothing new. In fact I was '*automating*' Citrix Metaframe builds using windows scripting techniques back in 2004. The market though, has become awash with different automation products and it's fair to say that many enterprises now suffer from '*automation sprawl*'. This results in a tactical rather than the strategic approach to automation required, meaning that benefits are never fully realized. In fact, I have spoken to many companies for which the word '*automation*' leaves a bitter taste in the mouth as they have been burned by failed implementations.

Now before everyone says "*John, what about solutions like Chef and Puppet? They have been very successful so far*" Yes, they have, I agree. They have both captured the market when it comes to turning infrastructure into code, so that environments can be rapidly and automatically built. I have no doubt that their solution or platform is easy to use but I would argue that the success of these solutions does not lie just with the product alone but with their execution. They made it very easy to share and encourage sharing of automation scripts and modules.

Here are my problems with Automation in relation to DevOps.

Problem 1: Speed can be dangerous
Speed is the nirvana for many enterprises when it comes to DevOps initiatives. The issue is that for many, their processes around release and change are centered on rigor and control, at the detriment of speed. I know I have sat through countless, pointless Change Approval Boards (CAB) in my time – one of the symptoms of this approach.

But going in all guns blazing in regards to release and change automation specifically, can damage your business. A widely accepted stat, backed up by formal research at Gartner, is that 80% of business service outages (read – application outages) are caused by release, change and configuration processes. Therefore the need for speed could easily amplify this stat. The solution is to make sure that you proactively monitor what matters in regards to business services. This means the end-user's experience, the application itself through to the infrastructure workload, including the database. Our Application Intelligence platform provides this, hence why it's a perfect partner to any infrastructure or application release automation tools.

Problem 2: Automation success isn't about the tools but the people and process
I have heard two great comments in regards to automation in last couple of years. Firstly, "*Don't automate what you don't understand" and secondly, "A bad process automated, is still a bad process*". I think these comments summarize nicely the problems which enterprises face in any strategic automation or DevOps initiative. While the automation product may be simple to use, understanding what to automate and what effect this will have on the organization and its people is a different matter.
If you are going to be successful with automation and DevOps, it's essential to address how an automation script or series scripts/modules will interact with, or change your current processes – processes around which jobs, roles and emotions are centered. This becomes difficult in an enterprise, which has defined processes for areas such as release, change and configuration, plus all the politics which come with this. Any strategic enterprise automation initiative has to overcome FUD (Fear, Uncertainty and Doubt) amongst employees in which automation will change their job activities.
By the way, automation may well improve an employee's role but this does not mean that FUD will not be present initially. To overcome this barrier you need to think about the people and process first. Involving people in automation initiatives (I would even suggest shying away from using the word automation) from the outset and being as transparent as possible through each stage of the adoption is vital.

Problem 3: Automation is more than just infrastructure and application automation
In my discussions with IT professionals in regard to DevOps, I hear a lot of excitement, quite rightly, in regards to infrastructure as code, application release and configuration release automation solutions. But this should not be your only focus for DevOps. It's still essential to be able to respond rapidly to emerging application and associated infrastructure workload issues before they impact the end-user (customer or employee). This means that automated issue response or remediation is an essential capability that you should look for in any APM solution.

One of our driving principles at AppDynamics is to enable businesses to act fast. This means that we make it easy to run a response to an issue or event. Another key capability is that we help enable modern application architectures by providing cloud

The CALMS Framework For DevOps cont'd

auto-scaling. Our platform understands application demand via real user monitoring so it automatically knows when to scale up or scale down application components in a cloud-based environment. This ensures that the customer or employee continues to have great performance even during heavy utilization periods. This feature is essential for those organizations who have to deal with periodic events such as Black Friday or Cyber Monday.

Lean

Defining Lean

For many, and for myself, the basis of modern Lean principles come from Toyota and the Toyota Production System. If you want to read about this in detail then I would recommend The Toyota Way, by Jeffrey Liker, a book that I read a couple a years ago and enjoyed. Often referred to as Lean Manufacturing, the definition on Wikipedia states that it's 'a systemic method for the elimination of waste within a manufacturing process'. Since then Lean IT has emerged as an extension of Lean manufacturing and focuses on eliminating waste in IT services while providing increased value to customers or employees.

The Principles Of Lean Applied to DevOps

The Principles Of Lean Applied to DevOps The main principles of Lean focus on identifying value streams, which in the case of DevOps, are applications that form services provided to customers, employees or partners. From this, value stream mapping visualization techniques are used to break down and analyze identified value streams into steps or processes, from design through to delivery, so that waste activities can be identified and eliminated. For example, in terms of monitoring, this would be tools which are not providing value to certain processes in the value stream, thus inhibiting flow and therefore not promoting speed and quality, essential to DevOps.

Six Lean Essentials For DevOps And Application Performance Management (APM)

Understanding and optimizing value streams is crucial. But Lean has a number of other key concepts, which can be applied to your APM strategy in order to ensure that it provides value to DevOps adoption:

Understand how applications create value for customers, employees or partners. Lean defines a concept called 'Gemba'. In Japanese this means 'actual place' and it means to understand where value is created for customers. In terms of APM, it's about understanding the value that an application creates for its users. This is important, as you can't create a business case for an APM solution without understanding the business importance of the application you are going to monitor. In relation to DevOps (delivering at speed while maintaining quality) features such as Real User Monitoring (RUM) help safeguard the value created while AppDynamics' ability to identify Business Transactions in an application mean that all key interactions, which deliver value to the user, are automatically identified and proactively monitored.

An APM solution must be 'second nature' in terms of operation. Students studying the Japanese martial art of Karate memorize and practice moves that become 'second nature'. This concept has been adopted by Lean and is about reinforcing certain activities, patterns and thought. In DevOps, this will mean reinforcing culture characteristics such as 'fail fast, fail forward'. APM is an essential activity that should be reinforced in DevOps adoption, and it's vital that an APM solution is easy to use so that it becomes 'second nature' in terms of operation. For that reason one of our driving principles at AppDynamics is ease of use.

Eliminate waste, inconsistency and absurdity from your APM strategy. There are three categories of waste in Lean: "muda" is a physical waste (e.g. time, money, and resources); "mura" is the inconsistency or unevenness of process; and "muri" means avoiding absurdity and unreasonableness. For DevOps, removing these three categories of waste is critical for speed. But they are also critical for your APM strategy, as multiple overlapping monitoring tools in a typical siloed enterprise mean physical waste (licenses etc), an inconsistency in the way they are used, and ultimately absurdity as alerts are not representative of the business. Hence at AppDynamics we provide you with end-to-end visibility of application performance and have smart alerting which means alerts are always in the business context.

APM must act as a 'fail-safe' against application performance issues. "Poka-yoke" is a Lean principle that means 'fail-safing' or 'mistake-proofing'. Essentially this is what any good APM solution should provide – the ability to make sure that application issues are remediated before they impact the customer or employee. As explained in the chapter on automation, our platform makes it easy to run a response to an issue or event meaning that a problem can be remedied before it impacts the business.

Contextual dashboards are critical to APM strategy and DevOps. Data from any APM solution only becomes information and knowledge if it's displayed in context of the audience. The Lean principle of "kanban" means "visual signal" or "card" and is about making sure that visuals are used to allow teams to communicate more easily on what work needs to be done and when. In DevOps, this easy communication is central to speed while maintaining quality and it's also central to APM. A great APM solution should allow you to generate visual reports and dashboards quickly in order to

The CALMS Framework For DevOps cont'd

communicate information relevant for the audience.

APM should provide the basis for continuous improvement in DevOps. DevOps success means delivering quality applications and new features at speed, which either internally, support employee productivity, or externally, delight customers. Therefore to stay ahead of the competition, the Lean principle of "kaizan" or continuous improvement of your people, process and technology in regards to DevOps is critical. As a great APM solution provides visibility of end-to-end performance of an application then this should be a primary source of information for performance improvement. At AppDynamics, with our built-in application analytics features, it also means that we provide the ability to not only continually improve performance, but your overall software strategy.

Measurement

The next letter in the DevOps CALMS model brings us to M for Measurement. Focusing on the right metrics and measurements is vital if you are going to succeed with DevOps adoption.

"You Never Know Where You Are Going Until You Know Where You Have Been"

There are not many times in my career that I am going to be able to quote Will Smith, but in the "Will2K" video, Will states "You Never Know Where You Are Going Until You Know Where You Have Been". For me, this summarizes why measurements and metrics are so important. Unless you can understand where you are right now then there is no way to understand whether you have been successful in regards to an initiative, in this case, DevOps adoption. Choosing the right DevOps measurements is important to eliciting the right behavior amongst IT teams, ensuring that they know what they need to focus on and, key to making sure that you are delivering the right value to the business.

Eight Essential DevOps Measurements

There are actually more than eight measurements that I would recommend for DevOps adoption. This is because your full measurement strategy really depends on what your stakeholders want to see, your goals and timescales for DevOps adoption. But at a top level, I believe there are eight metrics, which are essential, grouped around people, process and technology.

People measurements

In my experience most measurement strategies in regards to IT based initiatives focus on technology, but it's ultimately your focus on people that will lead to success. This means employees and the customer. Two key metrics here are:

Key employee retention – There will be employees in your team and your business that are key to DevOps success. Those who understand that DevOps is about taking an outside-in approach, focusing on business outcomes and delighting customers. Also those who can combine Development and Operations skills. These are skills you will want to hire, develop and retain. DevOps is a hot area from a job market perspective so you will want to make sure you keep talent in your business. Therefore identifying those employees who are vital to your DevOps efforts and then tracking retention is a must.

Customer satisfaction/experience – Ultimately DevOps success is not about fast release, collaboration or automation etc. Rather it's about making sure that software fueled business services satisfy and delight customers. This means that tying DevOps to customer satisfaction or experience measurements is vital. One option here is to link your DevOps measurements with NetPromoter scores if your company uses this popular customer satisfaction metric.

Process measurements

From a process perspective, the essential metrics which focus on release and change are:

Release/change frequency – One of the key drivers for DevOps adoption is to release new features or applications quicker than before. The aims here are to make sure that your application users (customers or employees) receive the value they need, are happy with functionality and ultimately stay loyal to your business. So measuring release/change frequency is essential. It could also be said that release/change frequency is one lagging metric for the leading measurement of customer satisfaction/experience.

Volume of failures – This measurement goes hand-in-hand with the measurement above. There is no point being able to release quickly if these releases fail and have to be rolled back.

Time/cost per release – Ok, maybe this should technically be two metrics. Time is money and money is required for a successful business. This means that it's critical to measure the time taken and cost per release especially in regards to Ecommerce based applications. If the cost of frequent releases outweighs the profit brought in by the application, then your DevOps adoption is not generating the business benefits that it needs to.

The CALMS Framework For DevOps cont'd

Technology measurements

There are a number of technology metrics that you may choose for your DevOps adoption but here are my three essentials:

Mean Time To Resolution (MTTR) – As mentioned, ultimately DevOps is about making sure that software strategy drives successful business outcomes by providing applications that help to deliver the service customers want. As such performance or availability issues are a DevOps strategy killer, and focusing on keeping MTTR measurements as low as possible is important. To improve, an end-to-end monitoring solution, which monitors end-user experience through to the application database or data store is critical.

Mean Time Between Failure (MTBF) - This metric comes from software engineering practices and focuses on making sure that applications are available. In the software-defined business highly available applications are key as they fuel business success. Therefore this metric is important BUT

Mean Time To Customer Impact – The reality is that failure only becomes an issue when it impacts a business process or the customer. Also failure as per the culture chapter is not a negative in DevOps as the mantra should be fail-fast, fail-forward. A good APM platform will help you to isolate and automatically remediate emerging application issues quickly before they impact the customer. Therefore focusing on Mean Time To Customer Impact is an essential metric for DevOps and monitoring efficiency.

Sharing

Sharing Is Caring With DevOps

In this final chapter on the DevOps CALMS model, we end on S for Sharing. Now I have to be honest, the first time I saw the CALMS model I initially questioned whether 'Sharing' was really needed. I mean, surely with the right 'Culture', then this would ensure sharing anyway? Even in the chapter on culture, I mentioned the need to shift from organizational silos to a collaborative model – which must include sharing, right? Well yes it does, but sharing (or collaboration) is really the core value of DevOps adoption. Without sharing, you can forget about having the right culture, your employees accepting automation, embracing Lean and having measurements that work. This is how important Sharing and collaboration is to DevOps success.

So What Do We Need To Share?

It would be easy for me to answer 'everything' but this is not going to help. In the software-defined business, where apps are key to commercial success, then it's essential that every participant in the planning, designing, building, deploying and supporting of apps is able to share information about their area/responsibilities/tasks and to be able to communicate, without friction, with each other. It must be easy to **feed back** information to teams involved earlier in this process and also **feed forward**. In the DevOps world these are referred to as **feedback** and **feed-forward** loops. The following diagram hopefully conveys this better than I can put into words:

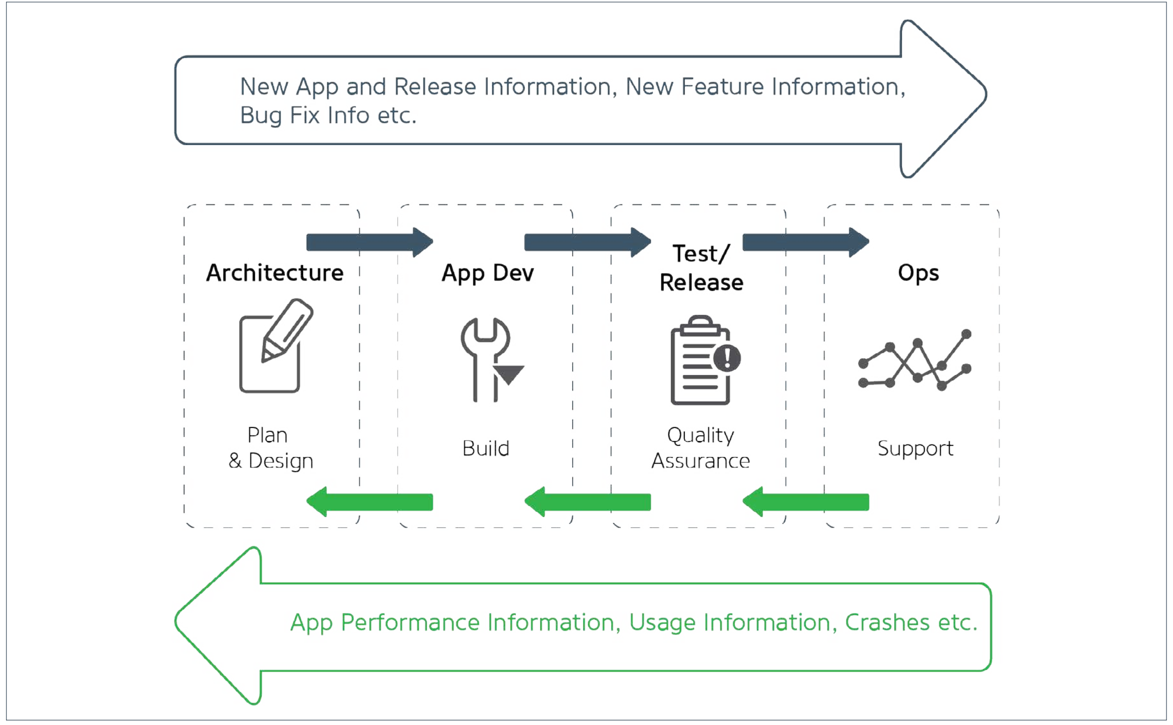

New App and Release Information, New Feature Information, Bug Fix Info etc.
Architecture
Plan & Design
App Dev
Build
Test/ Release
Quality Assurance
Ops
Support
App Performance Information, Usage Information, Crashes etc.

Figure 1 Feedback and feed-forward loop

App Performance Monitoring And Feedback

To share the information highlighted in the diagram above requires mechanisms and appropriate solutions – and application performance monitoring is a necessity here. It's a necessity along with other solutions that remove friction in communication, such as HipChat, amongst others, that promote easy sharing and collaboration.

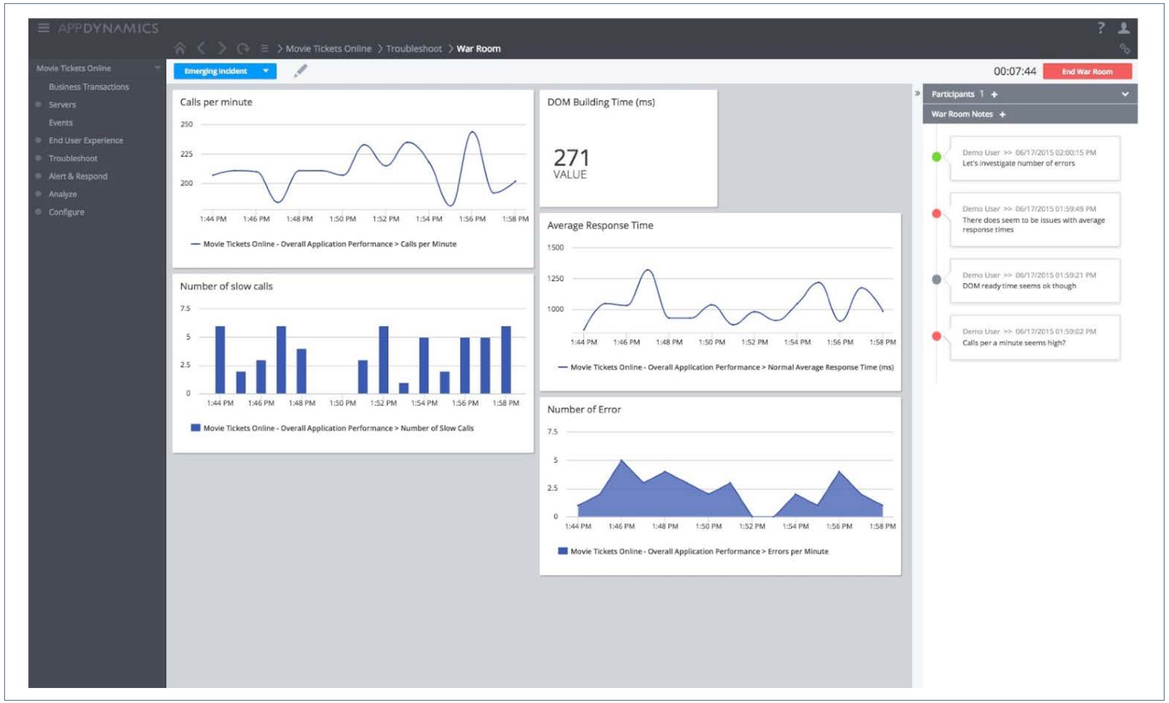
Without a good APM tool it's near on impossible to feed back, accurate and timely data about how applications are performing in production; information on end-user experience; and information relating to application code issues. APM is essential in helping your organization to run faster.

The CALMS Framework For DevOps cont’d

In fact a great APM solution should provide the software-defined business with situational awareness. This is a concept, which comes from US Marine training and refers to the fact that Marines on the battlefield need timely and accurate information to make the next decision and the next move. This is what a great APM solution should provide for your business, dev, test and operations teams – timely information to make the next decision in order to enhance software strategy.

The Virtual War Room

At AppDynamics we have gone a step further in order to promote collaboration in the enterprise, with our virtual war room feature that allows business, dev and ops teams to collaborate on an emerging app issue. The war room allows everyone to see the same screen, add and create charts and graphs that update in real time. Importantly we also provide the ability for participants to communicate with each other by an integrated chat function. This means that everyone can be looking at the same information or same source of truth, which helps speed up Mean Time To Resolution (MTTR) of emerging application issues.



Demo User >> 06/17/2015 02:00:15 PM
Let's investigate number of errors

Demo User >> 06/17/2015 01:59:49 PM
There does seem to be issues with average response times

Demo User >> 06/17/2015 01:59:21 PM
DOM ready time seems ok though

Demo User >> 06/17/2015 01:59:02 PM
Calls per a minute seems high?

Figure 2 Virtual War Room

Sharing Means Going Beyond DevOps To BizDevOps

While sharing information inside IT, so from Architecture to Operations is important, at AppDynamics, we believe that it should go beyond this. In the software-defined business every function is accountable for software success. Therefore it’s essential for feedback and feed-forward to and from line of business teams in marketing and product management. Again at AppDynamics we make sure that the information from monitoring and from our application analytics solutions is easy to be shared with anyone in the business.

So for successful DevOps adoption you need to focus on BizDevOps and the feedback and feed-forward loops should look more like this:

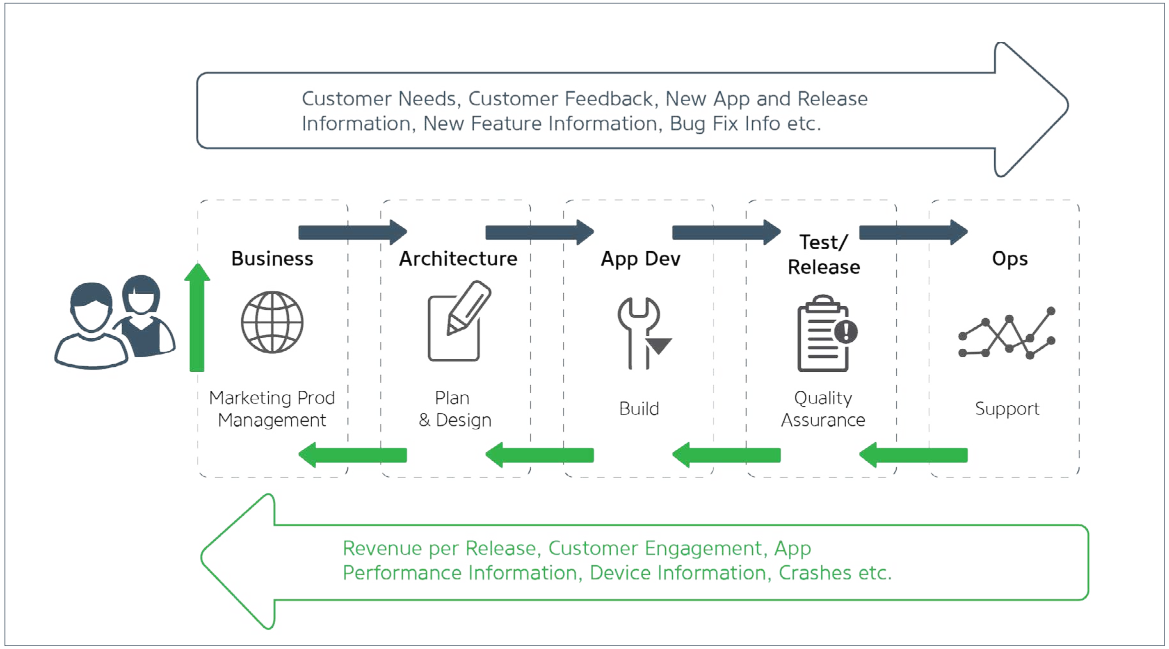

Customer Needs, Customer Feedback, New App and Release Information, New Feature Information, Bug Fix Info etc.
Business
Marketing Prod Management
Architecture
Plan & Design
App Dev
Build
Test/ Release
Quality Assurance
Ops
Support
Revenue per Release, Customer Engagement, App Performance Information, Device Information, Crashes etc.

Figure 3 Proper feedback and feed0forward loops

APPDYNAMICS