

Modern Monitoring 1

Actionable Guidance for IT Operations
and DevOps Practitioners



Foreword

Welcome to the very first edition of ***Modern Monitoring***, a collection of articles and insights designed to help IT operations and DevOps professionals deliver more resilient, supportable and high-performance IT services.

It's perhaps a sign of the times that monitoring as a discipline is receiving much more attention within the biz tech community. And deservedly so. The new distributed application architectures being built, together with the dizzying pace of software delivery, demand new approaches in what's traditionally been perceived as a "keeping the lights on" IT practice.

New monitoring approaches to address complex challenges is a common theme across this edition. As well as exploring the intentional chaos introduced by modern systems and how this impacts monitoring, we look at what technologies and techniques teams can adopt to increase capability—for example, how to use big data and analytics to gain more actionable insights into operational performance.

A key goal of Modern Monitoring is to equip professionals with guidance they can immediately use to learn and develop. That's why many of the articles focus on the strategies needed to move beyond fire-fighting and on-call stress-outs toward a more proactive discipline—one that helps people grow and benefits the entire business. For example, we explore how teams can better support testing in production with application performance monitoring, and what to do to put an end to post-traumatic alert fatigue.

Of course, there's no better way to consider monitoring than to draw parallels with other practices in related fields. It's why we've included a couple of pieces with a distinct aeronautical flavor that discuss importance of instrumentation and contextual awareness.



Contents

The DevOps Supremacy: How IT Operations Can Become Like Jason Bourne.....	4
Taming Complexity With Application Monitoring, Analytics and OODA Loops.....	7
Flight School for Application Performance Aces.....	10
Not Drowning, Waving: Four Ways to Survive the IT Operations Big Data Deluge.....	13
The Truth Is Out There: Application Monitoring in a World of Intentional Chaos	16
New Model Monitoring: How to Stop On-Call Burnout and End Post-Traumatic Alert Fatigue	19
Leveraging Cloud Services to Make In-House Ops Teams More Competitive.....	22
How to Better Support Testing in Production.....	25

The DevOps Supremacy: How IT Operations Can Become Like Jason Bourne

OK, call me an action nut, but I love the Bourne movies. Great stories, fast-paced, edge-of-your-seat thrills—brilliant entertainment.



But apart from the action, what I find fascinating is watching the uncanny abilities of the protagonist, Jason Bourne; his sixth sense, if you like. He can quickly “weigh up” his surroundings and make detailed assessments of all dangers. This acute level of awareness, plus, of course, his kick-ass fighting skills, give him, well—a distinct competitive advantage.

We need this ability in IT operations, especially with performance monitoring. We need Jason Bourne’s situational awareness.

Situational awareness is a digital business survival skill

As the term suggests, situational awareness means knowing what’s going on around you. It’s taught to U.S. marines, fight pilots, law enforcement officers and, yes, covert operatives. But what has all this to do with IT operations—well, actually more than you’d think.

Like Bourne movies, today’s digital business is equally intense. It’s a case of working “outside-in,” quickly understanding the needs of your customers and working faster than your competitors to deliver a high-quality experience. Now more than ever, it’s about anticipating and responding to any impending issues that could compromise this experience versus just being obsessive about finding the cause of problems.

Some folks in operations would argue they have great situational awareness, since they can monitor all the infrastructure and components underpinning applications. But is this good enough? Does monitoring at a detailed diagnostic level mean you’re paying attention to the right things from the perspective of the business and what your customers are experiencing in this very moment as they engage and interact with your business? Maybe if you have Spider-Man’s tingling spidey-sense abilities, but probably not if, like the rest of your team, you’re having trouble processing all the alerts constantly bombarding your plethora of monitoring tools and consoles.

Understand and assess customer experience—then act quickly and decisively

While observing infrastructure performance is important, it’s probably not enough to claim true awareness. What we also need is orientation, or knowing what information to look for when monitoring and then putting this into context of the all-important customer experience. Only then can we take actions that benefit the business.

“Not having infrastructure control, and apps literally being in the hands of customers, means **fixing problems** must give way to detecting impending failures. IT operations must step **out from behind the production curtain** and become custodians of the customer experience.”



But how do we know what to look when we're faced with an ever-growing number of events from a new generation of custom apps? How do we put all the diagnostic outputs into context so they become meaningful from a business and customer perspective? Like Bourne, this involves IT operations becoming agile and proactive. That's never achieved by waiting for production problems but by working collaboratively across the software lifecycle to anticipate any performance problems that will impact customers—right from the inception of a new business idea to complete software deployment.

So, thinking about Jason Bourne, what are some analogous situational awareness strategies we can adopt in IT operations?

- **Start thinking beyond simple baselining.**

Watch Bourne in action: He's always scanning his immediate environment, working out what's "normal." In IT operations, we work the same way. For any particular customer-facing application, we start with baselining—establishing normal performance in a given situation. This is fine when load is predictable, but now mobile apps, the cloud and uncertain demand make "normal" application performance much more difficult to ascertain. It's important, therefore, to replace "gut feel" baselining with more modern methods that can sense and respond to any given situation,

continuously recalibrating "normal." This not only prevents alert fatigue but also exposes hidden issues that could be damaging the customer experience.

- **Gain insights by analyzing performance behavior.**

Once we've developed a true picture of the performance landscape, it's important to quickly pinpoint all types anomalies—those issues that do not happen and should, or those that do happen and shouldn't. Here, analytics play a critical role in exposing valuable patterns from a vast amount of data, which is especially useful when presented in context of a user role or activity. For example, mobile app analytics should have the depth to provide business analysts visibility into where, how and when applications are being used by customers and how performance impacts this experience—a customer-centric or "outside-in" approach.

- **Enact responses where they're most effective.**

Engaging customers at scale with mobile and cloud is a wake-up call for IT operations. Not having infrastructure control, and apps literally being in the hands of customers, means fixing problems must give way to detecting impending failures. IT operations must step out from behind the production curtain and become custodians of the customer experience.

DevOps means IT operations becomes agile operations, defining new processes to better engage with development during build, test and deployment. It also means ensuring monitoring feedback is shared and incorporated into every software iteration or agile sprint. In this way, customer experience is imbued into everything, rather than being something we attempt to bolt on when it's too late.

While IT operations isn't facing the same adversity as Jason Bourne, they must optimize the experience of customers as they interact via digital channels. Put yourself in the shoes of your customers, taking an "outside-in" approach to application performance. Only then can you truly collaborate DevOps-style to drive improvements in application quality—all without ever having to drop down a gear, or, like Bourne, fight your way out of trouble.



Taming Complexity With Application Monitoring, Analytics and OODA Loops

Monitoring today's apps and digital architectures is a lot like aerial combat. Just when you think you've got everything under control, another slippery piece of tech gets right on your tail. Today that's containers and microservices—a big headache for operational flight aces.



When organizations adopt containers and microservice-style architectures in production, systems become incredibly complex. For operations, it's a shock, because it means coming to grips with many new container tech nuances—plus letting go of the old monitoring rulebook, because, well, it doesn't work anymore.

Depending on architectural patterns, containers introduce many more moving parts and dependencies—meaning more system checks, events and alarms. Moreover, the convenience of empowering development teams with a disposable and immutable platform upon which to accelerate software delivery doesn't come without its own set of gotchas, not least being woefully unprepared for the increased rate of change and complexity. If not addressed, this inevitably leads to outages, more organizational stress, perhaps even backing away from a technology that's a business no-brainer.

To combat these issues, some organizations are taking a page out of the cloud-native book with design for failure methods and increasing redundancy at every level of the technology stack. Some are going further than resilience, developing autonomous microservices that can handle and exploit the complexity of containers in order for the entire system to become stronger over time. These are commendable practices, but again, they place more demands on monitoring systems.

With so much complexity, it's perhaps not surprising that analytics is now touted as the operational answer to the container complexity

conundrum. And why not—take truckloads of data, logs and time-series, sprinkle in a liberal selection of algorithmic fairy-dust, and hey presto, problem solved. Well, here's hoping, but before running down to the analytics store, it's worth assessing capabilities based on how they help teams make fast decisions in constantly changing and dynamic container environments.

Interestingly, there's a proven DevOps practice taken from aerial combat called [OODA loops](#) (observe, orient, decide and act) that's an effective way of assessing the efficacy of application monitoring and analytics. By mapping OODA to a production container environment, teams can build a good picture of what constitutes an effective monitoring strategy and where analytics is especially important.

Observe: Any half-decent DevOps site reliability engineer uses monitoring solutions to collect masses of data from as many sources as possible. In container environments, where everything's in constant flux and events unfold rapidly, the best analytics (like fighter pilots) are capable of correlating information from multiple sources, continuously asking questions about what's happening across all the hosts, containers and pods and looking at the impact on service—essentially distilling mega amounts of information into actionable views. Here, good APM analytics use proven statistical methods like [differential analysis and dynamic baselining](#) to filter out signals from noise—especially relevant for microservices, where dependencies can produce cascades of alarms.

"Analytics doesn't end with information capture. Once data has been sorted and processed, solutions should be capable of identifying the **exact problem** with as little noise as possible."



Orient: Analytics doesn't end with information capture. Once data has been sorted and processed, solutions should be capable of identifying the exact problem with as little noise as possible. Again, more advanced methods use analytics to gather evidence and guide cross-functional support teams toward resolutions. One perfect example is the assisted triage feature of CA Application Performance Manager (CA APM), which leverages graph theory and workflow to quickly orient teams toward solutions—again, especially valuable for container environments, where problems can be multifaceted.

Decide: Based on all the symptoms identified during OODA orientation, the best monitoring systems help teams decide what action to take to address an event. For increased value, systems should surface this information to staff best positioned to act upon it in the context of their work. A good APM solution, for example, would detect a performance problem

associated with a software build; a better one would incorporate pass/fail conditions into the process and allow cross-build performance comparisons, all directly accessible from a developer's workstation and fully integrated into DevOps workflows and practices (continuous integration is this example).

Act: In dynamic container environments, app monitoring and analytics must allow for fast action once decisions have been made. This means integrating monitoring and analytical horsepower into many more workflows and processes to increase range and impact—for example, using app monitoring analytics to determine which coding practices lead to the best performance outcomes and instantiating those across teams. Or integrating analytics with auto scaling to dynamically optimize workloads based on load predictions—with great analytics solutions, the improvement opportunities are endless.

Like fighter pilots, application monitoring solutions must process and respond at speed. Look for modern solutions that allow OODA loops to be implemented, and measure their effectiveness based on how quickly they can act on problems and drive improvements

To learn more about what it takes to be an IT operations analytics Top Gun, attend www.ITOA2Summit.com or download the [Monitoring Redefined: Digital Experience Insights white paper](#).



Flight School for Application Performance Aces

In the early days of aviation, pilots flew by the “seat of their pants.” With little in the way of instrumentation, they relied on sight and judgment. That was fine when the skies were clear and conditions good, but not so great for flying in clouds or fog, and things got, well, soupy.



Stuck in the haze with no points of reference, pilots were in a space where nothing behaved normally. If they sensed the plane was descending, they pulled back on the yoke to gain altitude, only to find the plane diving more steeply. Even when they believed the plane was level, indicators suggested a sharp turn.

Not surprisingly, the best course of action was often to bail out before the plane hit the ground.

The need for comprehensive instrumentation

Gut feel and instinct aren't great for flying planes in adverse conditions. This has something to do with the way pilots process information using visual and vestibular systems to figure out where they are in space. As it turns out, fluid movements within the inner ear canal can play all sorts of tricks. What's actually level flight might actually be processed as a steep turn, and any intuitive action to correct the situation just makes matters worse. And without views of the horizon, pilots became so spatially disoriented that they lost complete control.

With the birth of "instrument flight" this problem was averted. It meant supplementing the basic navigational devices most aircraft carried with a cohesive set of instruments pilots could use to double-check what their very fallible senses were telling them. This included artificial horizons combined with turn and bank indicators.

But providing instruments wasn't a total solution. Pilots still had to become skilled in using them—and learn to trust them. It's why today's pilots can only fly in the soup when they've mastered the instruments.

Up in the soup—piloting modern applications

So what does any of this have to do with modern monitoring? Well, there are quite a few parallels.

In the not too distant past, monitoring was like the early days of aviation. Like the pilots of yesteryear, we mostly enjoyed clear visibility, and failure conditions across on-premises applications were predictable. But today's cloud-based applications are far less visible, and distributed systems behave in ways that challenge established monitoring beliefs. For example, a CPU utilization increase for a cloud app might now be a good thing, and any associated alerting becomes unnecessary and costly overhead.

So just as aviation involved an instrumentation leap of faith, the same must happen in monitoring. The systems we design and build must be capable of sensing, analyzing and responding to complexities mere mortals can never comprehend.

But this involves so much more than acquiring another tool or building fancy dashboards. Like modern avionics systems, modern monitoring must be capable of correlating information across various components and delivering what singular

"In the not too distant past, monitoring was like the early days of aviation. Like the pilots of yesteryear, we mostly **enjoyed clear visibility**, and failure conditions across on-premises applications were **predictable.**"



dials and widgets can never provide: context. It's easy to say, of course, but tricky to achieve.

The problem with more dials and dashboards

Too often as engineers we put faith in our own dashboarding systems. We build and maintain excellent systems for monitoring and alerting individuals of individual data points—CPU metrics for sysadmins, memory leaks and traces for app developers and so on. The only problem is that any action is driven without context. So while a network alarm seems troublesome to a network specialist, does it warrant immediate attention if customer experience isn't impacted? We don't know, because we lack context, without which we make the wrong decisions—like “bailing out” from a release when we don't need to, or blissfully ignoring slow-acting performance twists and turns that cumulatively bring a service to its knees.

In aviation, pilots are trained to quickly assess and correlate information across multiple readings on their instrumentation panel, never putting complete faith in one readout. But this isn't so easy in IT operations. Important data might be located in multiple repositories, held in different formats, even owned by other teams.

What we need are modern application monitoring systems that deliver the equivalent of what modern pilots provide, only fully automated and at massive scale: systems that automatically collect, scan, process and correlate data, be

that logs, metrics, transactions, whatever, across everything contributing to customer experience and better business outcomes—applications, infrastructure and networks.

Beyond monitoring: context is king

At CA Technologies, we recognize the importance of providing both contextual awareness and guidance. Take, for example CA APM, which now provides the capability to visualize and correlate infrastructure metrics with logical application topology. This means that when customers conduct Web performance monitoring across modern containerized applications, they have immediate access to host and container metrics—in context of application performance—all from one intuitive interface.

And with the rich insights context delivers, we can drive better actions; it becomes our automated guidance system. That's why many of our solutions provide automated workflows to not only analyze data and pinpoint likely root cause, but also intelligently support corrective remediation or performance optimization. One such example is the assisted triage functionality introduced with CA APM.

We can no longer manage application performance using gut feel and instinct—by the seat of our pants. We need solutions that deliver the full performance picture, whatever the complex technology soup, fog and cloud we find ourselves working in.



Not Drowning, Waving: Four Ways to Survive the IT Operations Big Data Deluge

The days of managing monolithic-style applications running on a single platform are well and truly over. Organizations are now committed to delivering their customers a far richer variety of digital services using multiple channels. This means applications are more likely to execute from the cloud, via a multitude of microservices interacting with virtualized resources, containers and software-defined networks.



In this new normal, teams can no longer afford to get bogged down with reactive fire-fighting and lengthy war room sessions. But with so many moving parts, increased application complexity and dizzying rates of software delivery, what strategies can IT operations employ to prevent being wiped out by waves of operational big data?

The traditional approach is to buy more monitoring tools, one for every new wave of technology adopted. But this doesn't scale, negatively impacts margins and only provides narrow views into the all-important customer experience. So, putting tools aside, where should organizations turn?

Well, to the data, for starters—or, more important, to the business problems and opportunities they solve and uncover.

This is hardly an epiphany. Web scale companies understand implicitly the importance of data and gleaning valuable insights. By developing analytics-driven applications, implementing at scale and democratizing usage, these businesses continuously raise the bar in terms of productivity, agility and customer engagement.

In a DevOps context these businesses thrive because their IT teams are equally analytics-driven. Not only do they surpass today's expectations for delivery speed and quality, they leverage data insights to drive improvements at every stage of the digital service continuum. So before perusing the extensive tools catalog, stop and consider four higher-value strategies.

1. Build an analytics-driven culture within IT

Many teams will collect masses of data points; however, what characterizes a strong analytics-driven culture is a focus on collectively leveraging metrics for the benefit of the business as a whole. This will involve:

- Empowering IT teams with the applications needed to uncover and share more powerful insights. These can include changes in customer engagement via new mobile app designs, emerging performance/security anomalies or optimum cloud architecture patterns.
- Incentivizing teams according to business performance goals and outcomes; avoiding persistent “vanity metrics” and operational outputs.
- Fast, real-time action or recommendations when insights are uncovered. Nothing demotivates teams faster than finding something valuable and then not being able to act on it.

2. Democratize data and analytics across the entire business

Analytics has limited value when only used by IT operations to support their daily grind. Better methods and techniques treat data as an enterprise asset many teams can use, share and leverage in a variety of different contexts. This could involve:

- Delivering an “analytics as a service” operational function where teams can build their own monitoring dashboards and reports to quickly gain the insights they need.
- Ensuring every group is provided with analytical models that have production-level support, together with real-time data that's ready to use and of known quality.
- Analytics-driven monitoring applications that immediately surface performance insights in the context of different roles and tasks.
- Rapid insight prototyping, which when shown to have value quickly becomes established in production processes and tools.

“By correlating many metric types across these elements (time series, logs, etc.), analytical models become a **shared mechanism** that DevOps teams use to **drive improvements**”



3. Start using analytics where they're most effective: customer experience

You can't manage what you can't measure, but collecting and measuring data that's truly reflective of customer experience is tricky. While some individual metrics will work in specific situations, it's more likely that combos, mashups and new derivations are needed.

In order to gain customer experience insights, analytics-driven applications will deliver teams complete understanding over cloud and on-premises application infrastructure, application performance and the underpinning network. By correlating many metric types across these elements (time series, logs, etc.), analytical models become a shared mechanism that DevOps teams use to drive improvements—for example, using predictive models to assess the business outcomes of new code based on application performance or latency improvements.

It's always possible, of course, that teams can revert back to a narrow (albeit analytical) perspectives to attack the problem. Some teams will be metrics-driven; others will use logs. The trick is understanding how a dovetailed approach will yield greater value—for example, correlating log analytics with network performance management for faster, accurate root-cause determination.

4. Become an effective hunter-gatherer: take a unified, system-level approach

There're no disputing that many IT organizations have invested time and money in acquiring a great set of tools to collect data. So why replace them? What's really needed is a scalable, open method to aggregate and normalize millions of metrics and logs into one unified data store. Call it an immutable Data Lake, Analytics Warehouse, whatever—having a centralized store of data helps teams quickly search, locate and visualize valuable trends, patterns and correlations. Without this, different groups may resort to managing their own (often overlapping) data sets, using inconsistent access methods, tools and data formats. That's fine for a narrow view but woefully inadequate when teams have to relate data across multiple data stovepipes.

As these strategies illustrate, being analytics-driven is so much more than fixing problems faster. When organizations invest in building an analytics culture, enact new customer-centric methods and share deep insights, the focus shifts toward treating every problem, pattern and anomaly as an opportunity to improve customer experience.

So stop getting wiped out by waves of data. Start using analytics-driven applications and surf toward a brighter business future.



The Truth Is Out There: Application Monitoring in a World of Intentional Chaos

< If you're a sci-fi nut you've probably checked out *Blade Runner 2049*. The seminal 1982 original is still my favorite, although the new movie ticks many boxes. Sure, it's a dark depiction of a dystopian future, but I love all the cyberpunk tech and references to artificial intelligence. >



"Together with all the modern tech, leading organizations are also employing **modern agile organizational practices** that better align previously fractured and siloed teams around business goals."

There's one AI nugget that I really enjoyed. It's the part where K (the main protagonist, and Blade Runner) has to undergo a rapid-response baseline test after any traumatic event—like "retiring" a rogue replicant. In his first test, K passes with flying colors, but later he fails—he's way off his baseline. In each test his responses seem consistent, but the machine conducting the test detects behavioral anomalies, which for a replicant are, well, life-threatening.

This all got me thinking that managing app performance across today's complex tech is similar. Detecting "normal" application performance is becoming much harder. Unlike in the past, when we had time to find and fix performance problems across apps that behaved in predictable ways, modern distributed systems are much less forgiving. Containers can be replaced in seconds, and apps can consist of thousands of independent microservices. What used to be considered bad—for example, high CPU utilization—might now be expected, even desired. It's confusing, chaotic, counterintuitive.

A map without context is now next to useless

To gain visibility into application topology, we've always relied on our trusted maps. This was fine when representing fairly static topology with a known number of tiers. But in a world of ephemerality and immutable infrastructure, it comes up short. Even if we can capture all the containers and dependencies, how do we represent them with limited console real estate? Where do we drill down to gain more clarity? The simple answer is, we don't know—the topology is too dense.

But in *X Files*-speak, "The truth is out there." We just have to take what's chaotic and convert it into something that makes sense and has purpose. This is both a challenge and an opportunity. It's challenging because we've tended to view mapping from a narrow perspective. Since ops has controlled the apps and the infrastructure, they build the maps. That's great for production support, but with increased application abstraction and empowered development, these views

have limited scope. They also lack an essential ingredient: context. Even if we could distill these new dynamic environments into palatable views, would they be useful for an app component developer who just wants to understand how her new code impacts performance in a software build? Maybe not.

Chaos is intentional—don't fight it, work with it

As it turns out, all this complexity provides an opportunity to do some wonderful things, but only if we work counterintuitively, turning to what we fear most—the chaotic application componentry itself. This means leveraging the descriptors or attributes available to fuel a new data model that supports multiple constituents and use cases.

And there's so much we can unlock. Docker, for example, provides a raft of attributes that my colleague Amy Feldman discussed in [a great article](#) outlining how these can be leveraged with a multi-dimensional data model to deliver different views or perspectives to many teams.¹



And because a component can have many attributes—like location, AWS availability zone, service owner, code version, build number—we can present the data in context of roles and functions, pivoting when needed. So as an app support manager I can visualize performance from a service-level perspective, while over in the next cube my app development colleague can be using the same solution to visualize application performance in context of her code.

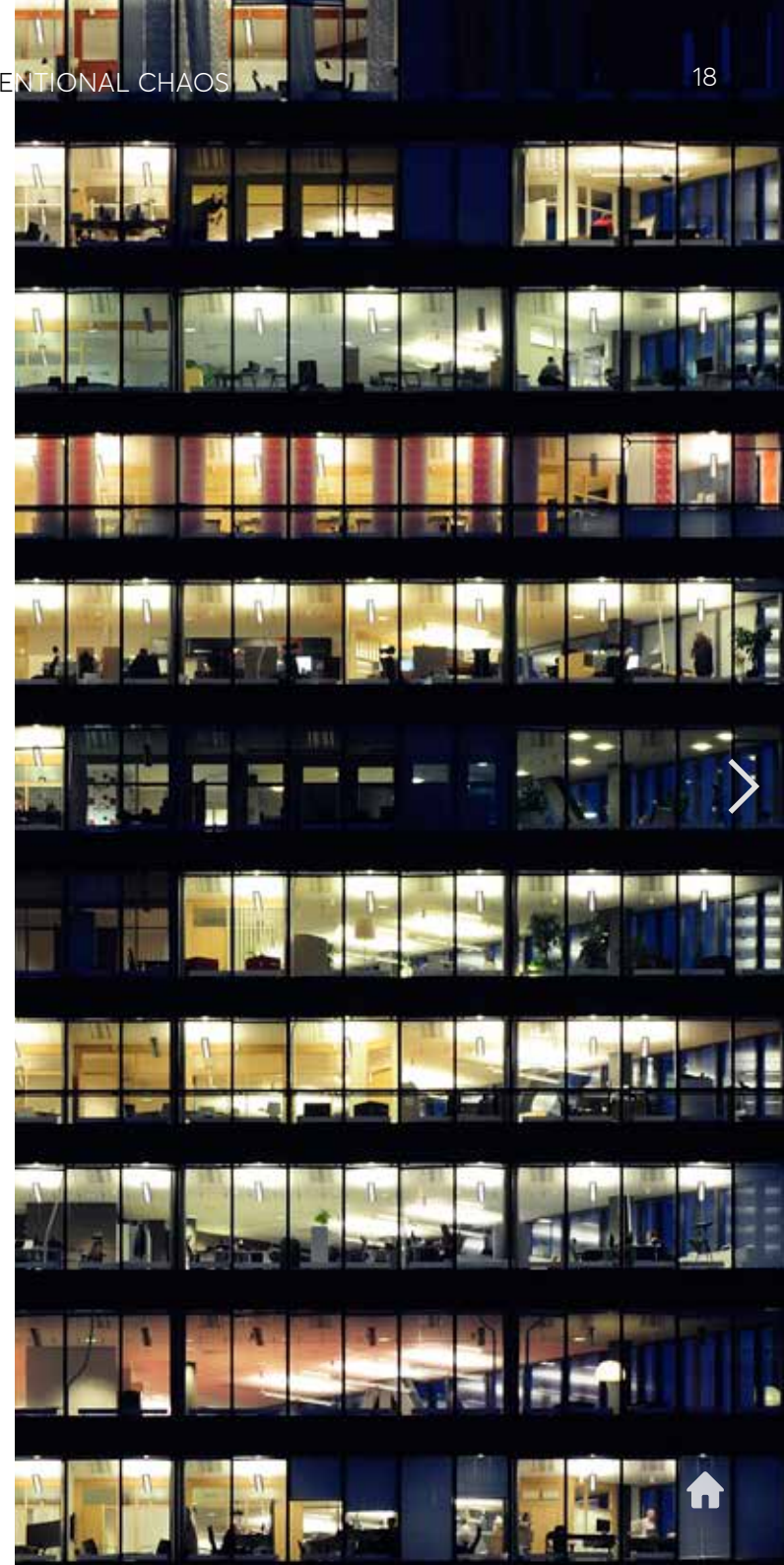
But we can take this much further.

Elevate your monitoring practice to a whole new level

Together with all the modern tech, leading organizations are also employing modern agile organizational practices that better align previously fractured and siloed teams around business goals. Rather than have separate dev and ops teams organized by tech function, they fuse these into vertical units responsible and collectively accountable to specific outcomes. Each of these could contain front-end developers, DBAs, sysadmins and so. Going further, many organizations also form horizontal cross-team groups where specialists (like front-end developers) get together to share ideas. Spotify, probably the best exponent of this model, goes even further, organizing communities (or guilds) that have common interests irrespective of an individual's own specialization.

Now imagine that as we reorganize we can leverage the data model described above to surface application performance insights that immediately reflect the outcome-driven structure. As specialists get together, they can share relevant and valuable information. This could be a mobile app dev group sharing which coding practice they've used to drive better performance—what worked, what didn't and why. Or perhaps it's a team of cloud architects comparing performance trends and anomalous conditions across different microservice deployment patterns, determining where the business is getting the most value from its investments. Call this agile, DevOps, whatever. To me it's a system enabling people do awesome things for their organization—and helping them grow as they do it.

With all the complexity and chaos, it's easy to get spooked. But before hitting the panic button, always remember, "The truth is out there"—somewhere. You just need a modern APM solution to find it—one backed by an open and flexible data model.



New Model Monitoring: How to Stop On-Call Burnout and End Post-Traumatic Alert Fatigue

In so many ways, IT operations has developed a military-style culture. If we're not fighting fires, we're triaging application casualties. We're the troubleshooters and problem solvers who hunker down in command centers and war rooms. Even when apps are stable, we reminisce over periods of conflict by recounting our "war stories."



Yes, IT operations is the kick-ass A-Team ready to leap into action. The best of the best, right?

Sadly, no. Because unlike the '80s TV A-Team, tech engineers actually get hurt.

In business tech, war is hell and no one wins—especially not the on-call staff who are constantly being dragged out of bed at 3 a.m. to engage in the technology skirmishes and firefights. For these battle-weary tech professionals, having to constantly deal with flaky infrastructure and poorly designed applications carries a heavy personal toll. Not only do they have a lousy work-life balance, they quickly come to hate the on-call role—and perhaps even the teams and organizations they work for. This can result in reduced staff morale, more AWOL and even, ahem—mass troop desertion.

So what are the signs an IT organization is engaged in bad on-call practices? There are three obvious ones to consider:

Support teams are overloaded. Any talk of continuous delivery counts for squat if systems are badly designed, hurriedly released and poorly tested. If teams are constantly running from one problem to another, someone or something will eventually break. Of course, good application support engineers try to do the right thing by patching up systems to keep them in action. But such are the stresses of working in these environments that no time is ever available to work up permanent solutions. The result: applications with Band-Aids just limp from one major outage to the next.

"In so many ways, IT operations has developed a **military-style culture**. If we're not fighting fires, we're triaging **application casualties**."

Bad practice becomes the norm. If on-call staff are constantly being asked to deal with floods of false alarms, any sense of urgency in responding to those alerts will be diminished. Staff become desensitized; [deviance is normalized](#). It's a problem well understood in the field of [healthcare](#), where clinical staff have been known to dial back cardiac alarm systems due to a nuisance factor. Similarly, in IT, when on-call staff have alert fatigue, they might be inclined to rejig some alert thresholds or hack up an automation to put old incident paging systems into snooze mode. Whatever the cheat, the results are never good.

Poor visibility and insight. What could be worse than being woken up at 3 a.m. to deal with a tech crisis? Answer: being woken up at 3 a.m. and being absolutely powerless to do anything about it. Even with a swag of open-source monitoring tools at their disposal, including log aggregators and dashboards systems, on-call teams still struggle to address complex

problems. Not because these tools are bad per se, but because narrowly focused monitoring only provides partial answers. That's always been troublesome, but now it's even more problematic due to the distributed, API-centric nature of microservice-style architectures.

Poor visibility doesn't only manifest technically; there are people issues too. If senior managers aren't aware of on-call burnout or just turn a blind eye, methods should be employed to help them wake up and smell the stink. A good place to start is discussing the people cost associated with stressful on-call rotations. If, however, the empathetic approach falls short, try presenting all those latency, saturation and utilization issues in the context of business impact—like revenue, profit and customer satisfaction.

Apart from using monitoring to present on-call calamities in clear business terms, there are a many other common-sense approaches that can help give on-callers their life back.



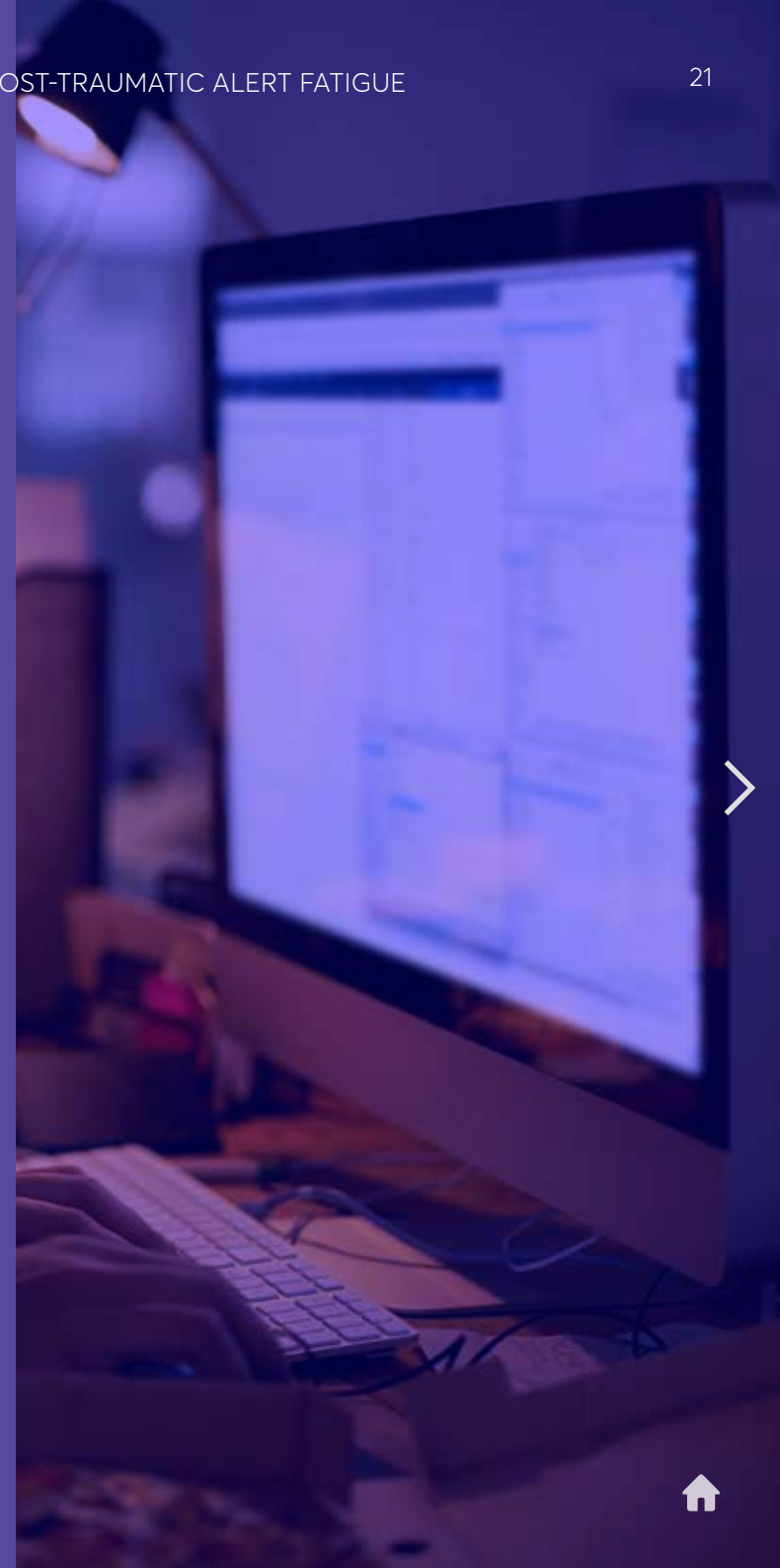
Make alerts actionable. What's the point of alerting on machine-related issues when they have no tangible impact on the business? Good monitoring avoids this by aggregating metrics at a service level and only alerting on-call staff when customers are hurting and problems need fixing immediately. Anything else can wait until tomorrow, when everyone's had a good night's sleep.

Automate runbooks. It's a good practice to develop concise documentation that guides on-call staff during major service disruptions. That's all fine and dandy, but runbook effectiveness is highly dependent on development teams providing clear and up-to-date instructions, which isn't always top of mind. Although there's no substitute for good support documentation, advanced analytics-based monitoring tools can augment manual detective work with fully automated evidence gathering, correlation and recovery workflows.

Put developers on call. However good on-call support engineers are, no one knows the idiosyncrasies of an application better than the people who wrote the actual code. Putting developers on call means the people who most likely caused the problem are the ones being put on the spot to fix it. Witnessing programming stuff-ups firsthand in the small hours of the morning is also a great motivator to put things right—permanently.

Audit continuously. Even if the ultimate goal is to never page on-call staff, a more realistic objective is to ensure staff never get paged for the same problem twice. Again, good monitoring tools and analytics can support this, by, for example, reviewing performance alerts over historical time periods and correlating with any infrastructure or code changes.

When staff are continuously placed in stressful on-call situations, they'll burn out—and so will your business. With constant on-call reviews and auditing together with advanced monitoring practices, organizations can eliminate alert fatigue, increase service reliability and reduce the need for costly unplanned work.



Leveraging Cloud Services to Make In-House Ops Teams More Competitive

The mighty, magical interwebs are awash in metaphorical majesty about the benefits of a strong DevOps culture and healthy collaboration. Often, much of that information stops just short of saying this: These benefits derive from empowerment—specifically, empowering the people on operations, business and creative teams to make their own decisions.



Real empowerment delivers real benefits. When teams are trusted to make their own decisions, they'll usually do what's best for them without negatively impacting the business over the long term. Since business value today derives from speed of delivery, the choice of tools with which teams work is becoming more local, more exclusive, less enterprise-driven. Ease of use, avoiding vendor lock-in, expediting project readiness and promoting openness are key factors in each of these decisions. This way, new mandates for features or new requirements can be quickly addressed with new solutions or by swapping out.

Perhaps the best thing about these more local development environments is the "winners are grinders" factor. When teams continuously adopt innovative technologies to solve real and immediate problems, other teams will gravitate toward the solutions they find. Call it DevOps-style collaboration, DevOps culture, whatever—people will back the winners rather than continue to uphold centrally maintained decisions, rigid policies and calcified practices.

Today, IT operations are faced with a barrage of questions about their relevance and viability in the enterprise, especially since centralized IT management services are now easily obtainable from cloud service providers. These new services are perceived—correctly or not—as easier to access, simpler to use and more effective.

How should IT operations teams respond? Perhaps the only viable response is not with argument but with action: Build and deliver integrated services that ease the burden on

those tasked with delivering quality software at speed and scale. Give developers easy access to observability and monitoring services that can profile the performance impacts of their coding decisions before deployment.

IT operations teams should not be faced every day with hurling code over the castle walls into the production environment and ferociously defending their entrenched structures with rigid change management policies and standardization dictates—the boiling oil of enterprise IT. If these teams are given the option of competing, they take on the mantle of service provider and adopt some of their standards and practices as well—going above and beyond, treating colleagues as customers and crafting valuable services.

"Today, IT operations are faced with a barrage of questions about their **relevance** and **viability** in the enterprise, especially since centralized IT management services are now **easily obtainable** from cloud service providers."

In traditional production environments, application monitoring has been used in a break/fix context. In this new environment, monitoring becomes essential for increasing competitiveness, but it requires new approaches in terms of tools and practices:

1. Monitor to Communicate

Highly competitive internal service providers know that other teams—who are, after all, their customers—have unlimited choice. So it's important that they vigorously demonstrate the value of their service offerings through constant communication.

Here, modern monitoring can be highly effective. For example, when ops teams provide the entire organization with business service status screens, everyone has shared visibility into the health of their running applications. Ideally, teams can configure these services according to their own requirements, although the greatest value comes from providing a single source of truth.

Naturally, the truth isn't always palatable, especially when one is on the receiving end of it. No matter how good alerting systems may be, it's crazy for response teams to keep putting out the same fires. Therefore, modern monitoring services should always mix the good news with the bad news, without using the same brush to paint both. For instance, make use of analytical hotness to do a little "dashboard shaming" on all the repeat systemic offenders. This might sound painful, but fragile systems can only get fixed when the chaos they cause is made visible to management.



2. Targeted Marketing

Prospective customers for monitoring services won't necessarily be other operations teams—they could be developers, product managers, data scientists, even business analysts. These folks need simple explanations of what these services provide and how they personally benefit.

Trying to dump existing monitoring services, no matter how complex and unwieldy, will be fruitless. So ensure that ops teams provide these services with clear instructions, and that they deliver seamless integration. For example, with an application monitor rooted in Jenkins, developers can see the performance impacts of their code. An agentless Docker monitoring service may be marketed and packaged as providing immediate microservices performance visibility without painful configuration.

3. Extended Service and Support

With so many choices and alternatives open to their customers, internal teams must differentiate the breadth and depth of their services. Support is one potential differentiator.

Leveraging their built-in advantage of proximity to the customer's businesses, the best teams work upstream with their customers. Perhaps you've heard of [Gemba walking](#), where one interacts with users, listens to their ideas and their complaints, and offers empathy for their business pains. Less personal, more technical examples include a pre- and post-deployment monitoring snapshot service, or a fully integrated APM including a load testing function.

In any case, every offering should be backed with superior customer service and a great experience. That way, your service isn't just addressing problems but also helping forge strong relationships—the key ingredient of DevOps.

With more and more cost-effective alternatives available to the people they serve, an organization's internal teams must become competitively savvy about the services they provide. Building off of traditional production functions such as monitoring to produce tailored, seamless offerings that are differentiated by ease-of-use, customer service and support, everyone prospers.



How to Better Support Testing in Production

< The term “testing in production” is polarizing. Mention it to some teams and they’ll say it’s sheer genius, a critical method for gaining realistic insight into app performance behavior. Others might scoff that it’s just dev and QA playing catch-up—running tests they didn’t have time for earlier in the development cycle.



But testing in production is more about supporting the realities of modern software development than it is about playing catch-up. Yes, the term will probably get business folks and ops teams squirming, but in truth it's an essential element of successful DevOps and continuous delivery initiatives.

In the modern development world of APIs, cloud, containers, microservice architectures and the internet of everything, testing in production isn't just common sense—it's really the only way to progress. Massively complex distributed applications will perform in crazy ways, users will encounter problems you never anticipated, and cloud services no longer under your direct control can and will fail. And without a production testing strategy in place, these problems will challenge even the best operational capabilities. Worse, they'll remove critical feedback loops and negatively impact decision making.

But in truth, testing in production isn't new; it's been happening for years. Every time we slavishly update our Facebook pages and LinkedIn profiles, we're active participants in experimentation, even if we don't know it.

In enterprise IT, testing in production goes by many names and is also becoming well established. This includes:

- **Blue green deployments:** This is a practice that reduces downtime risk by running identical production environments (blue and green), only one of which is live (blue). As you prepare an app release for deployment, testing is

conducted in the green environment. Once you have the green light (forgive the pun), routing traffic is switched to the green system. It's not without its challenges, but it's a great way to roll back quickly when things go wrong.

- **A/B or split testing:** This is used to test new features of an application against a variety of factors, like uptake, popularity, conversions, and how these impact revenue and profitability. A/B production testing is mostly involved with the UI, but of course back systems need to be available and performant to do it reliably.

"But testing in production is more about supporting the realities of **modern software development** than it is about playing catch-up."


- **Canary releases:** As the name suggests, this practice involves sending out a version of your software (the canary) into the live environment (the coal mine) to see how it performs. Here you check critical integrations, response times, error rates, CPU, memory utilization, etc. If the canary app keeps chirping, great. If it dies, no great loss—you have the feedback to quickly roll back the deployment.

Irrespective of method, the one consistent requirement for testing in production is short and fast feedback loops. The faster you gain information about application performance, the quicker you can get out of Dodge by rolling back the release, or proceed and reap the benefits.

Modern application monitoring is absolutely essential for fast feedback, and CA APM provides many great features to support it. These include:

- **Visibility into KPIs and SLAs.** CA APM can be easily configured to monitor critical changes to KPIs for business services. Using Experience views, teams have a single line of sight into app performance, aggregating key performance metrics into overall health scores. With canary releases these provide indisputable evidence that your canary (app) is failing, but—and here's the rub—in the context of business outcomes and customer experience.



- 
- **Change impact:** the what, when and how. It's one thing to roll back quickly, but it's another being able to figure out what caused the problem. As updates are made—be they software, environmental or architecture—we need to ensure that the folks making the changes are alerted as quickly as possible. CA APM supports this in a number of ways.
 - With fully automated transaction tracing, developers can quickly pinpoint problematic code.
 - Using Timelines, architecture teams can track back to determine what specific environmental update caused degraded performance.
 - By layering infrastructure onto app topology maps, app support teams can quickly correlate symptomatic performance conditions with related infrastructure dependencies.
 - **Facing the “unknown unknowns.”** Complex distributed applications behave in complex ways. They can exhibit production behaviors never anticipated. In the past it was pretty straightforward to check and alert on known failures. Now, however, it becomes difficult to detect severe conditions that build slowly and cumulatively over time—they can fall under the radar.

CA APM's differential analysis can help address these complex issues. Employing proven statistical methods and analytics, it distinguishes nuisance alarms from anomalous trends worthy of attention. When combined with blue green deployments, it could provide a reliable and precise roll-back trigger, not least because every alert in these complex environments shouldn't force teams to hit the panic button.

This isn't a definitive list of how CA APM supports testing in production. Depending on your own organizational practices and use cases, there are many more applications.

For example, CA APM's open data model and Team Center could be employed to surface key attributes contributing to performance, comparing blue green deployments side by side. In an A/B testing scenario, the integration between CA APM and [CA App Experience Analytics](#) can provide some great insights, like, for example determining that slow customer uptake of a new feature is actually related to back-end performance, not the feature itself. Great insight leads to better decision-making.

Testing in production shouldn't be feared, castigated or reviled. It's an essential part of modern software development. To do it effectively, we need effective app performance monitoring.





Learn more about modern monitoring and DevOps practices at ca.com/apm

CA Technologies (NASDAQ: CA) creates software that fuels transformation for companies and enables them to seize the opportunities of the application economy. Software is at the heart of every business, in every industry. From planning to development to management and security, CA is working with companies worldwide to change the way we live, transact and communicate—across mobile, private and public cloud, distributed and mainframe environments.

1 Amy Feldman, "How Does Your Docker Monitoring Tool Visualize Complex Relationships?"
August 30, 2017, www.ca.com/en/blog-apm/docker-monitoring-tools.html

Copyright © 2018 CA. All rights reserved. All trademarks, trade names, service marks and logos referenced herein belong to their respective companies.

200-326896_0118

