WHITE PAPER • MAY 2019

ca technologies
A Broadcom Company

AIOps Essentials

A Unified Data Model as the Foundation for AIOps

Table of Contents

Executive Summary

Challenge

The highly complex nature of today's modern cloud-based distributed applications challenge monitoring approaches. To address this, DevOps practitioners and site reliability engineers (SREs) are increasingly exploring algorithmic IT and artificial intelligence (AI) as the means to detect and predict problems faster while prescribing automated self-healing and recovery processes. However, these goals are compromised by the need to maintain, manage, and integrate a multiplicity of tools—each with their own unique and separate data model.

Opportunity

By adopting a unified data model dynamically built using a time-journaled directed graph of attributed objects, teams have the analytical foundation upon which to collect, group, correlate, and visualize more complex performance conditions spanning applications, infrastructure, and networks. Flexible and open-ended, this modern approach allows every data source and condition to be analyzed and interpreted in the context of shared team goals and critical business outcomes.

Benefits

CA Technologies, A Broadcom Company, employs a unified data model that enables teams to gain complete visibility across modern application environments. Open, extensible, ontology-agnostic and using time as a primary dimension, this dynamic model allows teams to purposefully apply AI and machine learning to drive substantive improvements in application performance without sacrificing speed.

Significant benefits include:

- Determining which methods, practices, and architectural patterns correlate to the best outcomes.
- Gaining full-stack visibility into any combination of conditions impacting customer experience.
- Surfacing hidden insights without integrating fragmented models, data tagging, or making error-prone assumptions.
- Future-proofing of the business by seamlessly leveraging new technology data types and attributes.

Application Complexity Is the New Normal

The move toward an agile operation encompassing multi-cloud platforms, containers, and distributed application architectures, combined with DevOps and continuous delivery practices, challenges the effectiveness of traditional monitoring tools and practices. Such is the complexity of modern applications that many IT operations teams are now struggling to detect, let alone predict, the complex failure conditions and circumstances that can adversely impact performance. And since software applications are now inextricably associated with business success, failure to respond carries far greater penalties that are both sudden and swift.

As a result, monitoring as a discipline has begun to suffer. It has become much harder to keep track of complex applications and infrastructure comprising many moving parts while meeting more aggressive performance expectations and consistently delivering a superior customer experience. In response, what used to be adequate for older three-tier application architectures (with more predictable patterns and better understood error conditions) is proving to be woefully inadequate for today's modern software applications.

There are many factors challenging traditional monitoring:

- **Static to dynamic systems**. Development cycles are increasingly shorter and compressed, with teams even needing to conduct live experiments and gain hypothetical learnings against production systems. In such environments, fast feedback is key. Unfortunately the traditional approach of establishing monitoring as an add-on function slows delivery and fails to deliver more expansive insight.
- **Data to insights**. Businesses are increasingly demanding operational data to demonstrate and differentiate the quality of services delivered to customers, and as a means to justify new investments in future-proofing technologies. Many of these data elements are beyond the scope of traditional monitoring tools. When data is captured, it is normally held in separate silos. These silos make more proactive analysis time-consuming and costly.
- **More stakeholders**. The shift to agile development and DevOps-style collaboration increases the purview and expectations of monitoring. While monitoring used to be the purview of IT operations teams, development and QA teams now require monitoring data presented in the context of coding and testing processes. At the same time product owners seek business- centric key performance indicators (KPIs) based on the performance of critical applications.
- **Specialists to generalists**. New job categories related to application health and performance are emerging, such as monitoring architects, observability, and SREs. Focused less on discrete technologies and more on the performance of systems as a whole, these new practitioners require more effective data on the relationship between the cause and effect of problems eventuating across complex distributed applications.

These factors have raised the stakes considerably. There's renewed interest in monitoring, with much discussion around the failings of traditional approaches. Now, entire conferences are dedicated to monitoring (such as Monitorama and SREcon), and the number of new monitoring tools (including open source and commercial products) has increased considerably.

With regard to monitoring products, the tools themselves do an excellent job at collecting and generating substantial amounts of data using a variety of different models. This includes but is not limited to:

- Metrics, calculators, and alert generators.
- Distributed tracing systems.
- Synthetic transactions and real-user monitoring.
- Application and system logs.
- Logical application structures.
- System incidents, SLAs, and KPIs.
- Computing and storage infrastructure.
- Network topology, traffic flow, and status.
- Detailed response-time analysis and latency.
- Error rates and saturation.

As the complexity of systems increases, it is not surprising that new tooling methods will surface to address the idiosyncrasies of each new technology introduced. Take, for example, serverless computing, where the highly dynamic nature of the technology dictates capturing and analyzing a new set of metrics, such as functional invocation counts and cold and warm start times. Compare this to a microservice style application deployed across a 1,000-node cluster, with the need to measure overall service performance versus short-lived (ephemeral) containers, and another data model will emerge.

The problem isn't the lack of data that tools process or generate; it's an inconsistent approach to data modeling. While each tool is good in itself, it can only document and visualize individual piece-parts of a highly complex system. While this may be valuable in ascertaining technology performance conditions, the higher-level goal of understanding how overall business performance is impacted by a mesh of interrelated components and dependencies is well beyond its range.

Inconsistent data modeling results in a number of serious problems.

Contextual Blindness

Unless properly contextualized, each new data source adds a new costly overhead: conceptual debt. Consider a case where an alarm notification system has detected a problem with a network element. Most systems can reliably detect the problem, even determine the problem and visualize this against an infrastructure topology map.

However, because the model is maintained in isolation, it cannot unequivocally determine which services will as a result go out of compliance in terms of customer experience. This connection can only be made by presenting this data in context to another team using another tool based on a different data model.

Gap Stitching

What cannot be modeled cannot be effectively monitored, diagnosed, interpreted or predicted. Without this, users will be forced to address modeling shortfalls themselves. This can range from the rudimentary tagging of static data elements (such as before an application build process) or by attempting to integrate solutions, each of which has its own unique data model (for example, logical application elements with physical infrastructure). This is exacerbated further when fragmented and inconsistent models exist within individual products themselves, which is typical when monitoring functionality has been extended through hurried partner extensions and technology acquisitions, but without careful consideration for the increased operational overhead.

Bridging the Gap With a Unified Data Model

To address the problems discussed in this paper requires a unified data model. This will serve as a bridge needed to expose important monitoring data sources, contextualize that information and serve as the foundation upon which to more purposefully apply advanced analytics.

In developing this model, we have been guided by a number of important design principles, including:

- **Flexible and ontology-agnostic**. Be capable of understanding the complex properties and relationships within and across multiple data types, including logical and physical systems and the large volume and variety of data consumed and generated by monitoring systems.
- **Post-deployment-extensible**. Future-proof the organization by enabling new technologies and data types to be quickly incorporated within the model without enforcing the need to manually identify and tag elements. A truly flexible data model doesn't dictate rules; rather, interpretation must be left to the user.
- **Use time as a primary dimension**. Allow both performance changes eventuating from faults (infrastructure, software configurations, deployments) and seasonal fluctuations, together with structural changes (workload characteristics, before and after key milestones to be time-journaled). The ability to roll back the model is key for proactive analysis in constantly changing environments.
- **Highly contextual**. Every new layer of information and every new data source must increase understanding, and to do that it must be properly contextualized. For example, compute data utilization (CPU) exposed through infrastructure monitoring should be presented in the context of the application supported and their performance (and vice versa).

Unified Data Model Solution: Directed Graph of Attributed Objects

To support these goals, the unified data model and database serving as the foundation for analytics and machine learning in monitoring solutions been developed using graph theory-an area of mathematics that studies the nature of networks (in a broad sense) and complex systems through their connections and relationships. A graph data model is comprised of:

- **Vertices (also known as Nodes)**. The entities (often called nodes) of the model. Nodes can hold any number of attributes or key-value-pairs. They can have labels representing their different roles, which in addition to contextualizing node and relationship properties can also be used to attach meta-data.
- **Edges (also known as Dependencies)/Relationships**. These provide the relevant connections between two vertices. A relationship always has a direction, a start node and an end node. Although they can be directed, relationships can be navigated in either direction.

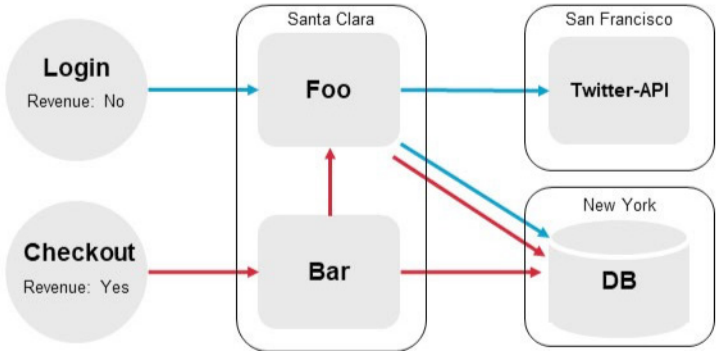
In the area of monitoring, graphs are an excellent and natural way to model complex IT networks, infrastructure devices, application logic, and their associations. Graphs enable organizations to connect multiple monitoring tools and gain critical insights within and across different elements managed by different teams. From dynamic dependency mapping to monitoring complex containerized microservices, the use cases for a graph-based data model are limitless.

By way of example, and as shown in Figure 1, a simple application is represented in a directed graph. The vertices represent application elements (transaction types, application servlets, database, and so on.), while the edge/dependencies show the relationship between them. Attributes for each vertex or node are shown in the table columns (type, data center, revenue).

FIGURE 1.
Directed Graph of Attributed Objects-Simple Application

Vertex ID	Type	Data Center	Revenue?
Login	Transaction		No
Foo	Servlet	Santa Clara	
Bar	Servlet	Santa Clara	
DB	Database	New York	
Twitter-API	API	San Francisco	
Checkout	Transaction		Yes

Edge (Dependency)	Context
Login -> Foo	Login
Foo -> DB	Login
Foo -> Twitter-API	Login
Checkout -> Bar	Checkout
Bar -> DB	Checkout
Bar -> Foo	Checkout
Foo -> DB	Checkout



While this is a rudimentary example, it illustrates the power and versatility of a graph data model. The extensibility of the model is demonstrated by incorporating attributes-revenue (associated with login and checkout vertices) and data center (apps Foo, Bar and DB). These and other attributes (programmatically or custom-applied) allow users to examine topologies from multiple perspectives, including organization and team structures, geo-location, and application priority. It is also important to note that the graph edge/dependencies in the model help increase the contextual understanding of the application, here, for example, showing how servlet Foo supports both the login and checkout transactions.

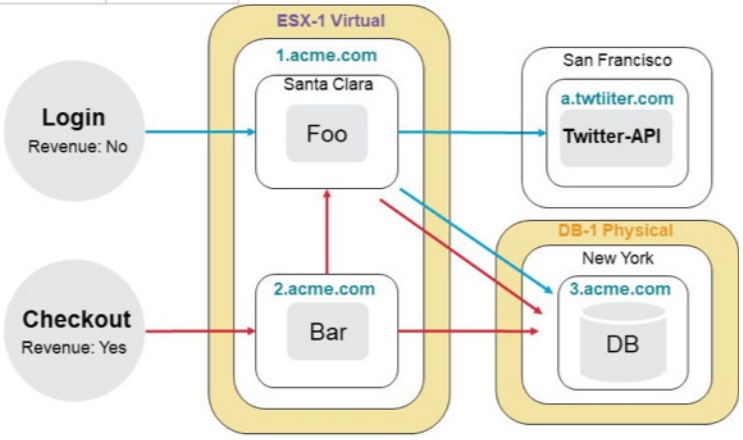
The power of graph theory is demonstrated further when data occurring at different layers of the application stack are layered onto the model as additional attributed objects. This is critical, since without this, administrators working at the application layer have no visibility into infrastructure dependencies. This traditional problem is resolved when the data model layers data at both the application and infrastructure layers (Figure 2). Now the directed graph-based model enables further understanding into how complex applications and related infrastructure interoperate, enabling teams to uncover previously hidden relationships and patterns. This can be extended further when more topological data providers add more data to the model (such as network elements and traffic). By exposing and layering elements across the stack, visibility increases, as does the ability to conduct accurate root cause determination, because problems often manifest at the deepest point.

FIGURE 2.
Direct Graph of Attributed Objects—Application and Infrastructure Layers

Vertex ID	Type	Data Center	Revenue	Host
Login	Transaction		No	
Foo	Servlet	Santa Clara		1.Acme.com
Bar	Servlet	Santa Clara		2.Acme.com
DB	Database	New York		3.Acme.com
Twitter-API	API	San Francisco		a.twitter.com
Checkout	Transaction		Yes	

Host	Type	Vendor	Name
1.acme.com	Virtual	VMware	ESX-1
2.acme.com	Virtual	VMware	ESX-1
3.acme.com	Physical	Dell	DB-1

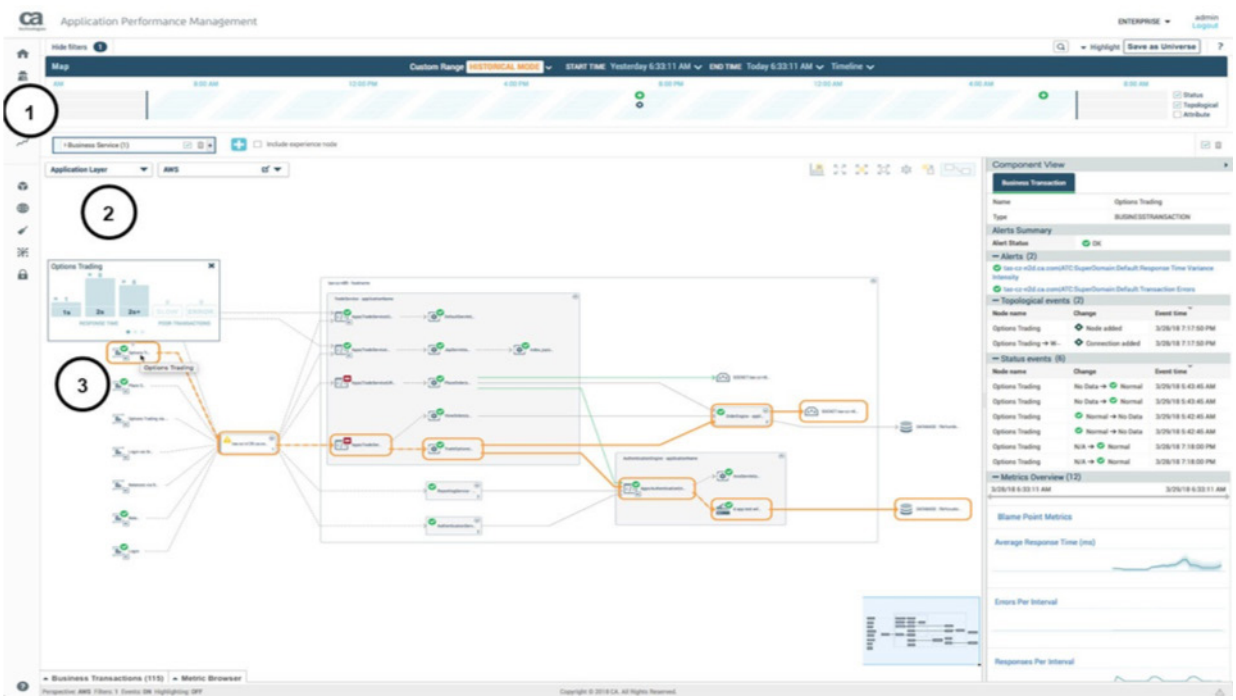
Edge (Dependency)	Context
Login -> Foo	Login
Foo -> DB	Login
Foo -> Twitter-API	Login
Checkout -> Bar	Checkout
Bar -> DB	Checkout
Bar -> ABC	Checkout
Foo -> DB	Checkout



As discussed above, our approach incorporates time as a primary dimension of the unified data model. With this capability, users can correlate performance changes across application and infrastructure layers. Since changes are time-journaled, users can roll back the data model to pinpoint performance conditions before and after major milestones. As shown in Figure 3, the DX Application Performance Management (DX APM) solution is shown leveraging the unified data model, layering and time journaling to:

- Visualize dependencies between application and infrastructure objects
- Query time correlations within and across different layers
- Examine monitoring data in context of role, task, activity, location (for example, AWS availability zones), container clusters, and so on.
- Determine attribute propagation through graph model traversal queries
- Understand performance conditions before, during and after critical events, including topology and architecture changes, DevOps pipeline application builds and rolling deployments

FIGURE 3.
Monitoring Application of Unified Data Model—DX APM



1. DX APM timeline functionality leverages time journaling to pinpoint status, topology and attribute changes.

2. Unified model allows logical (application) and physical (infra) objects to be layered. Attribute filters (in this case, AWS) can be applied.

3. Model supports graph traversal queries within and across app anc infra layers. Data is surfaced in presented contextually.

Analytical Application of a Unified Data Model

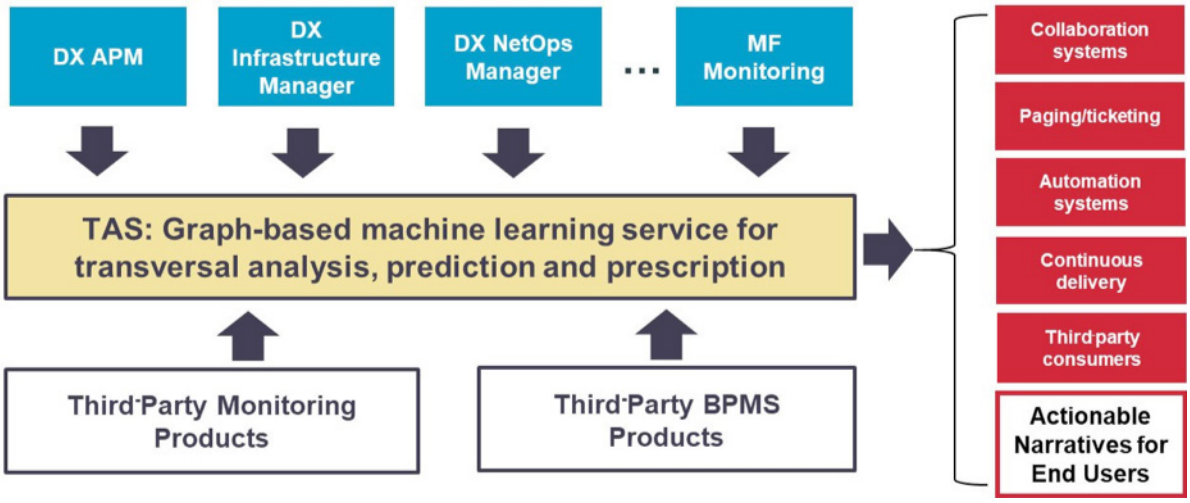
To exploit the benefits discussed in this paper, our extensive portfolio of monitoring solutions all project data onto the data model for the application of analytics (named Topology Analytics Service-TAS).

As shown in Figure 4, this service consumes and correlates data layers, including logical application topology information from DX APM, together with network and infrastructure information from DX Infrastructure Management and mainframe monitoring products. Such is the flexibility that many other sources from existing and future investments can be added. This further enriches the model and allows for holistic analysis that is representative of an organization's unique characteristics.

Of note too, is that the graph-based analytics service acts as both a consumer and a rich information provider. This is especially beneficial for the following processes:

- **Collaboration systems**. Change updates (such as code build) and the impact on performance can be shared and communicated across teams (such as through a Slack channel).
- **Paging, incident management/ticketing**. With contextual insight and traversal analysis, serious business- impacting problems can be prioritized and remediated faster.
- **Automation solutions**. Structural model changes analyzed across time and the associated performance are used as automation trigger points for deployments, rollbacks, auto-scaling, and so on.

FIGURE 4.
Topological Analytics Service



Underpinned by the unified data model that correlates related attributes across application and infrastructure layers, and as shown in Figure 5, TAS provides two essential analytics services. The services provided are Machine Learning Analysts and an Assisted Triage Engine.

Machine Learning (ML) Analysts

These are independent machine learning modules that access the data model and subscribe to certain types of events (business impact, resource conditions, errors/stalls and database events). Operating within specific domains, the role of ML analysts is to continually work to identify whether an event (such as a JVM memory leak) has eventuated or not. This is achieved by encoding domain knowledge (for example resource conditions: "death by a thousand cuts" and "pig in a python") within each analyst. As the event categorization "workhorses," analysts provide the triggers for effective root-cause analysis.

FIGURE 5.
Topological Analytics Service—Component Detail

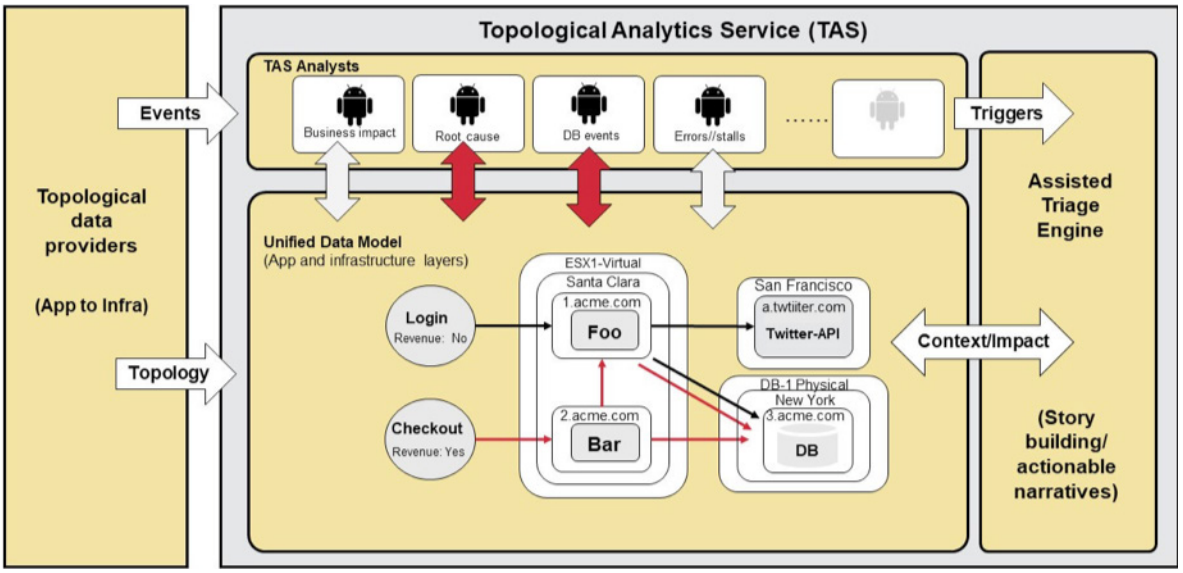
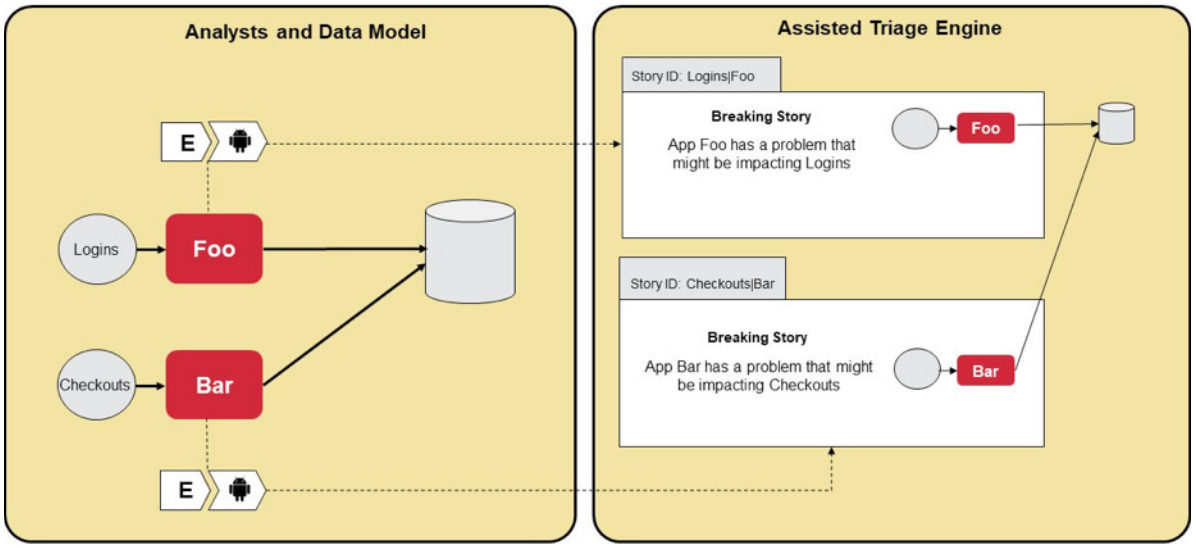


FIGURE 6.
Problem is Detected with an Application



Assisted Triage Engine (ATE)

By correlating relatable attributes within the data model and acting as a sandbox for ML analysts, assisted triage works to build stories or "narratives" on emerging conditions, their priority and context. Using narratives, users can quickly detect business-impacting conditions, pre-contextualize and group classes of events.

Importantly, narratives can also be used to build organizational knowledge and drive improvements-for example, serving as a key information source in runbooks to identify and remediate reoccurring problems.

To illustrate the narrative building capability of TAS (data model + ML Analysts + Assisted Triage), consider Figure 6 to Figure 8.

In Figure 6, a problem is detected with the application Foo, which starts to throw events or exceptions that match the criteria of a specific analyst.

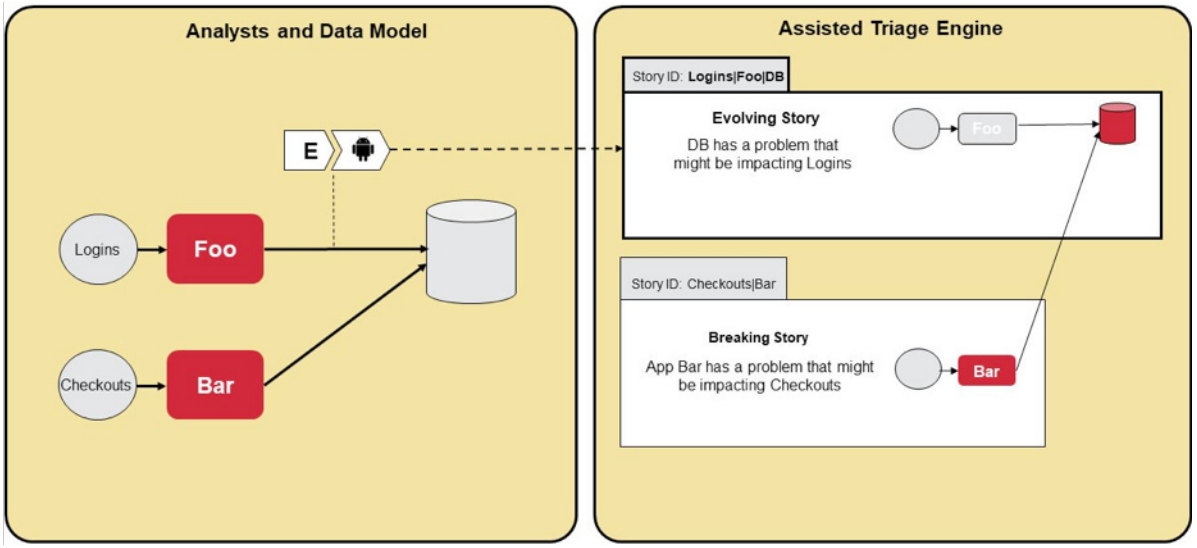
NOTE: Events can be simple/trivial baseline breaches, but we have also created machine learning modules that can be leveraged upstream to enhance events. These include:

- **Anomaly detection**. Including univariate, multi-variate, neural networks and random forest algorithms to identify items, events or observations that do not conform to an expected pattern or other items in the data model.
- **Dynamic baselining**. Using methods such as differential analysis (based on the western electric rules) and kernel density estimation to distinguish nuisance alarms and false positives from business-impacting events.

The analysts categorize events and trigger the ATE, which then maintains them as stories or narratives within a boundary (in Figure 6, the first "breaking story" is Logins | Foo-a problem has been detected that might be impacting Logins).

As also shown in Figure 6, another application (bar) catches the attention of another analyst. A new analyst participates because this application might be exhibiting a new behavior (such as response slowing versus throwing exceptions). Once again, ATE is triggered, and a new story (Checkouts | Bar) is written.

FIGURE 7.
Analyst Detects Events on the Graph Model Edge/Dependency



In Figure 7, an analyst with encoded domain knowledge on database connectivity also detects events on the graph model edge/dependency between Foo and a database (DB). Here ATE doesn't break out another story but enhances the existing narrative because the event is in the same execution path as directed by the data model.

Finally, and as shown in Figure 8, an analyst working at the edge/dependency between application bar and the database detects a problem. Once triggered, ATE determines that DB is impacting both applications and merges the two narratives into one (story ID: [Logins, Checkouts|DB]).

FIGURE 8.
Analyst Detects a Problem Between Application Bar and the Database

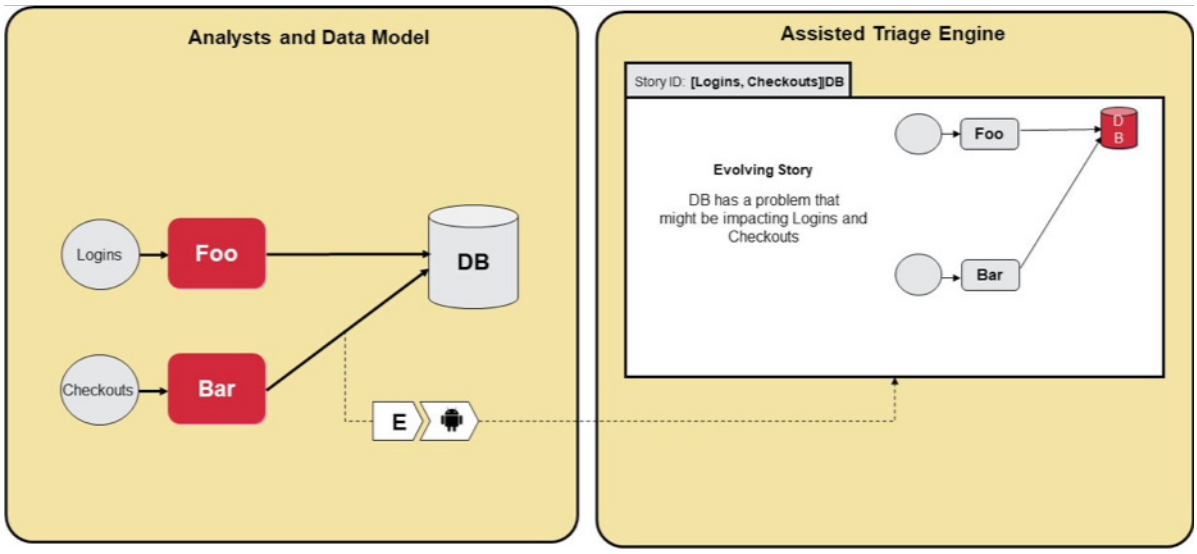
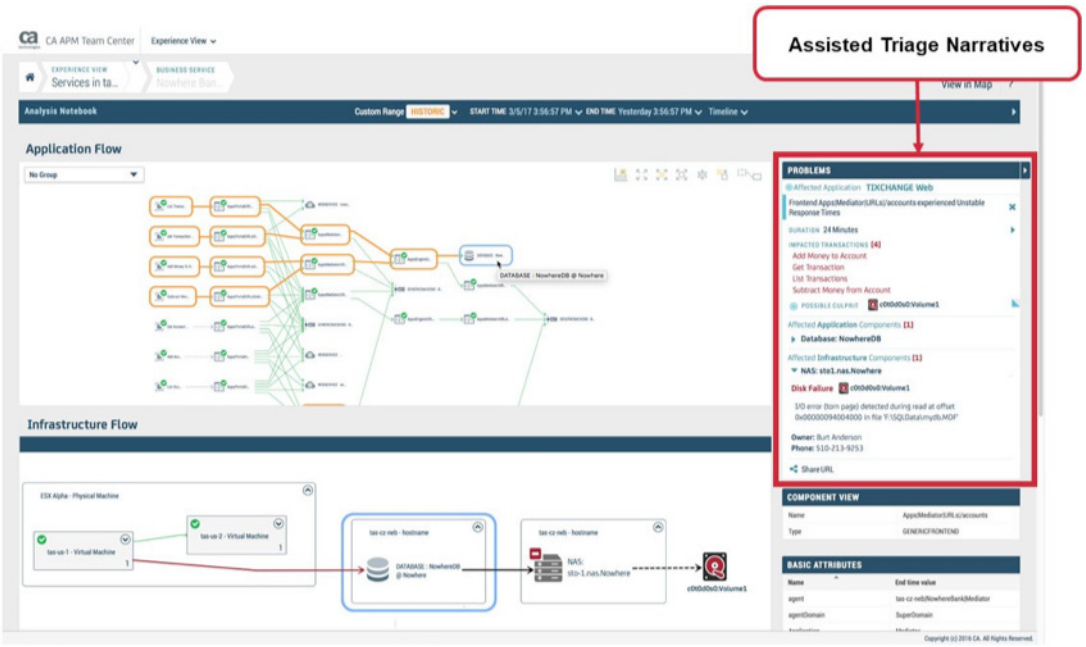


FIGURE 9.
DX APM—Single UI Projecting Data Model (App and Infra Layers) and Assisted Triage Narratives



It should also be noted that object attributes or key-value pairs provide the essential context needed for analysts and the ATE to prioritize business-impacting transactions (such as the revenue-bearing transactions of Bar over Foo).

While rudimentary, this example again illustrates how a graph-based data model unifying business transactions, applications and infrastructure elements is an essential foundation for analytics.

To be fully effective, the narratives or stories written by the ATE are presented as a human readable digest. These outline four things:

- What's affected at a transactional or service level
- The use-cases impacted
- How long the condition has been active
- The deepest software application or infrastructure element/culprit

Figure 9 illustrates how the analyst notebook functionality of DX APM has accessed the ATE to present all this detail from one user interface. The system presents the rich graph data (including objects at an application and infrastructure layer) and the ATE narratives. Using this interface, engineers can traverse topologies, filter data based on attributional values and quickly pinpoint the root cause of a problem.

Conclusions

Many IT organizations are challenged by the disparate data generated and stored in siloed and interoperable monitoring systems. Consequently, these organizations cannot easily access and share the data needed to quickly fix complex problems and enact much-needed quality improvements.

With the rise of modern architectures and the DevOps methods that are required to support digital business initiatives, it is imperative that organizations adopt an AI and machine learning approach to analyzing the massive amounts of IT operations data. However, to be fully effective and deliver maximum insight, analytics needs a unified data model to serve as the consistent foundation upon which data can be consistently consumed, combined and correlated.

By leveraging the directed graph of attributed objects approach discussed in this paper, organizations can quickly visualize important relationships, app to infrastructure, plus gain the comprehensive and contextual insight needed to purposefully apply and prioritize AI and machine learning. Furthermore, since data changes are time-journaled, organizations gain a more dynamic model-one that continuously surfaces key performance insights before, during and after key events, including workload, seasonal demand, and configuration and architecture changes.

This paper has also illustrated the benefit of using a unified data model in an AI and machine learning context by presenting our Topological Analytics Service, in which encoded ML analysts operating across the data model trigger a powerful assisted triage engine to rapidly and accurately pinpoint the root cause of performance problems wherever and however they eventuate, within and across both the application and infrastructure layers of modern technology stacks.

While this paper has focused on the application of a unified data model from a root-cause analysis perspective, it also serves as a foundation for advanced AI and machine learning methods across the analytics continuum, including detection, prediction and prescription.

For more information, please visit **ca.com/aiops**.

About the Author

An AIOps chief architect and former chief architect of DX APM, Erhan Giral has been working in the APM space for nine years and holds various patents around operational intelligence and application monitoring. Previously, he worked on various graph visualization and analytics frameworks, SDKs, and monitoring solutions servicing diverse industry verticals.