A MOVING TARGET: RETHINKING SERVICE MODELING FOR DYNAMIC WORKLOADS

In the news, we regularly hear about how changes or faults result in major outages at large corporations, impacting millions of end users (or thousands of airline passengers, in the case of the Notice to Air Missions (NOTAM) system outage earlier this year).

Changes or issues with downstream dependencies like applications, infrastructures, or networks can impact critical services, resulting in financial penalties and brand damage.

Traditional service modeling practices

Historically, configuration management databases (CMBDs) have helped organizations identify and resolve issues faster, manage risks associated with change, and make more informed decisions. Configuration item (CI) details are updated in the CMBD either manually or, more commonly, using a discovery tool that runs infrastructure and network scans on a periodic basis (daily or weekly in most cases). Manually maintaining CIs and their relationships has been a challenge. Most customers I have talked to regularly express concerns regarding the accuracy of their CMDB.

Service modeling is used to fence off targeted CIs to define a service boundary that represents a business, application, or technical service. Over time, users have modeled thousands of such services, which have proven difficult to maintain and keep current based on the shifting IT

landscape.

Service models help organizations identify critical business services and their downstream dependencies in order to measure service risk and fault. When enriched with artificial intelligence (AI) and machine learning (ML) insights, models that represent a critical business service should help mitigate outages and prevent bad headlines; however, service modeling processes and technology need to evolve and take modern workloads into consideration.

The challenges created by modern workloads

Modern applications, infrastructures, and networks are software-defined, virtual, and ephemeral in nature. CIs and their relationships to the modern IT landscape change very frequently based on cloud-scale workloads. DevOps practices introduce new features daily, and cloud architecture can scale on demand depending on user load; this helps bring new features to market faster, optimize resources for cost savings, and improve the customer experience.

This continuous change, however, poses a major challenge for traditional service modeling practices. Discovery tools continue to play an important role in this modern landscape for asset discovery, change impact analysis, and dependency mappings; however, they are not enough. A dynamic approach to service modeling is required to deal with near real-time changes in hybrid-multicloud environments.

Moving to dynamic service modeling

Enrich service models with CIs from monitoring tools

Monitoring tools provide additional CI details and their relationships at a higher frequency and, in the case of real user and application performance monitoring, do it based on real user transactions, making it more accurate (refer to Figure 1 below for additional details). For example:

- **Real user or customer experience monitoring tools** provide end user experience details with geolocation; this helps overlay service health and impact with customer experience metrics (e.g., click-through or bounce rate for websites or delay and jitter for voice).
- **Application performance monitoring tools** provide application-specific details on software components and their relationships across containers, cloud, distributed, and mainframe topologies. This provides an accurate and up-to-date snapshot of an end-to-end user journey, from mobile to the mainframe.
- **Infrastructure and network monitoring tools** provide low-level details and connect virtual and physical devices. These tools complement discovery tool scans.

Monitoring Tools	Frequency of Scans/Updates	Change Frequency
Real user monitoring	< 1 min	High
Application performance monitoring	< 1 min	High
Infrastructure performance monitoring	1–5 minutes	Medium
Network performance monitoring	5–30 minutes	Medium-Low
Discovery infrastructure/network scan	Daily or weekly	Medium-Low

Figure 1. Scan and update frequencies associated with different monitoring tools.

Take advantage of a highly performant graph database

Traditional CMDBs use relational databases for storing CI details and their relationships, which addresses asset and change management use cases. However, to account for modern operational use cases, there is also a need for a graph database to operate at scale, account for changing CI relationships, and maintain a history of the moving landscape.

Figure 2 below shows a typical CI landscape with siloed layers. There is a need to reconcile discovered CI relationships across the application, container, infrastructure (cloud, distributed), network, and mainframe layers. This is a hard problem and has been discussed in detail in my recent blog post, "How BMC HelixGPT-Powered AIOps Connects Observability Silos for Faster Probable Root Cause Isolation."

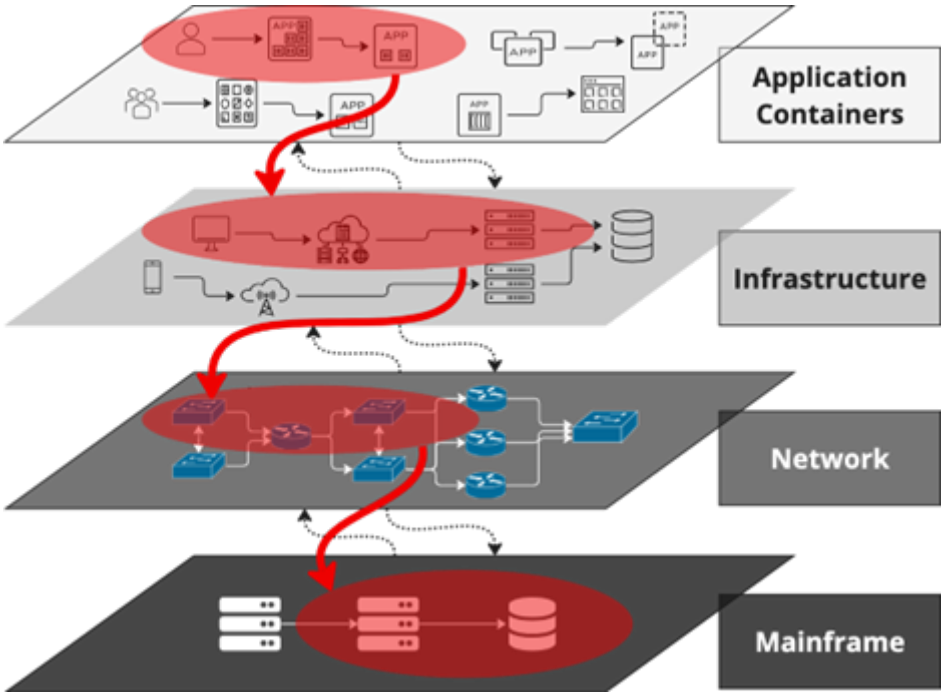

Figure 2. Layers with reconciled topology and service CIs shown in red.

Fortunately, the next generation of service modeling datastores are purpose-built to handle highly complex and connected data structures and use highly performant graph traversal algorithms to query data.

Next, model your business service

Assuming we have a dynamic and reconciled graph database of our IT landscape, the next step is to model a business service, where the goal is to connect all the CIs representing the service as shown in red above. This is a three-step process:

1. Identify one or more applications used by end users. If there is an application performance monitoring tool, then that is the best starting point. Provide application details as the inputs to the service modeling tool.
2. Dynamically traverse all these layers to automatically stitch together all dependent CIs for the application service. This requires a reusable blueprint that dynamically filters and pattern-matches to build the end-to-end service. Our example shows end users connecting to cloud-based and distributed applications that eventually connect to a database on the mainframe (see Figure 3 below).
3. Identify end user key performance indicators and use them to calculate the service health score. A problem with a network switch port flapping is only critical, for example, if it is

impacting an end-user experience like response time or voice quality.

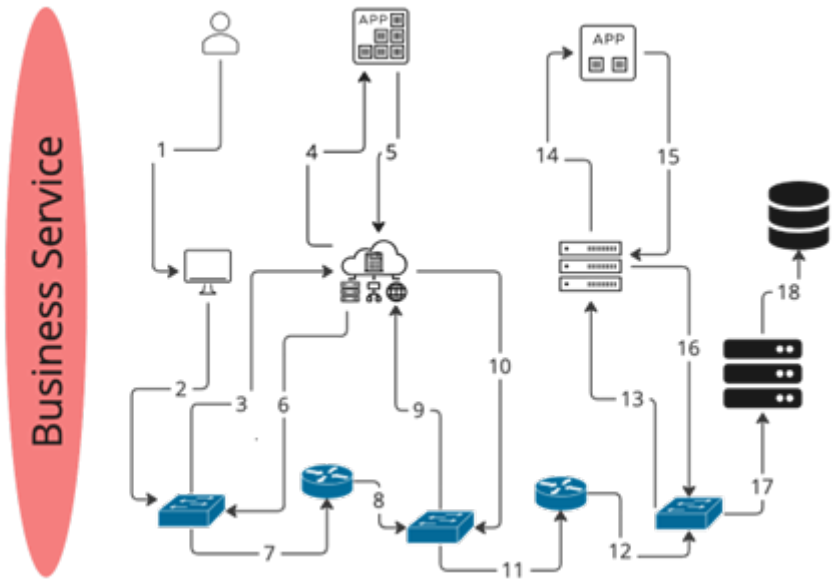

Figure 3. Business service with connected topology.

Model a business service using reusable blueprints

Blueprints are reusable, can be created and maintained using a graphical editor, and can apply predefined patterns and filters to a connected topology stored in the graph database. Blueprints understand vendor- or industry-specific CI relationship mapping. Unique topology models exist for monitoring and infrastructure solutions like AppDynamics, Dynatrace, SolarWinds, VMware, Kubernetes, OpenShift, and mainframe, etc.

Figure 4 below shows an example blueprint for an AppDynamics and Kubernetes deployment. The AppDynamics blueprint starts with an application name and connects software components to virtual/physical hosts and the underlying network devices, whereas the Kubernetes blueprint starts with a namespace and then connects to deployments, pods, clusters, hosts, and network devices.

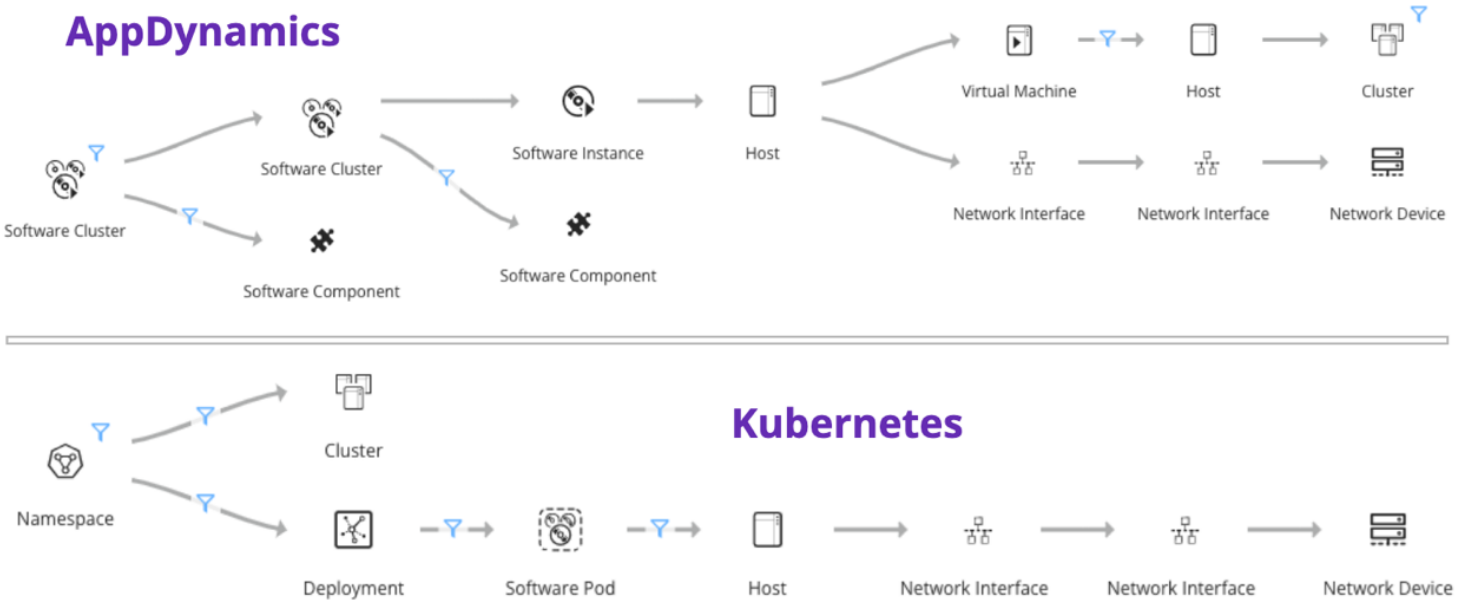

Figure 4. AppDynamics and Kubernetes blueprint example and reconciliation with virtual/physical infrastructure and network devices.

Figure 5 below shows the topology from the AppDynamics monitoring tool, which is limited to

application-only topology and has no knowledge of the underlying physical infrastructure and network.

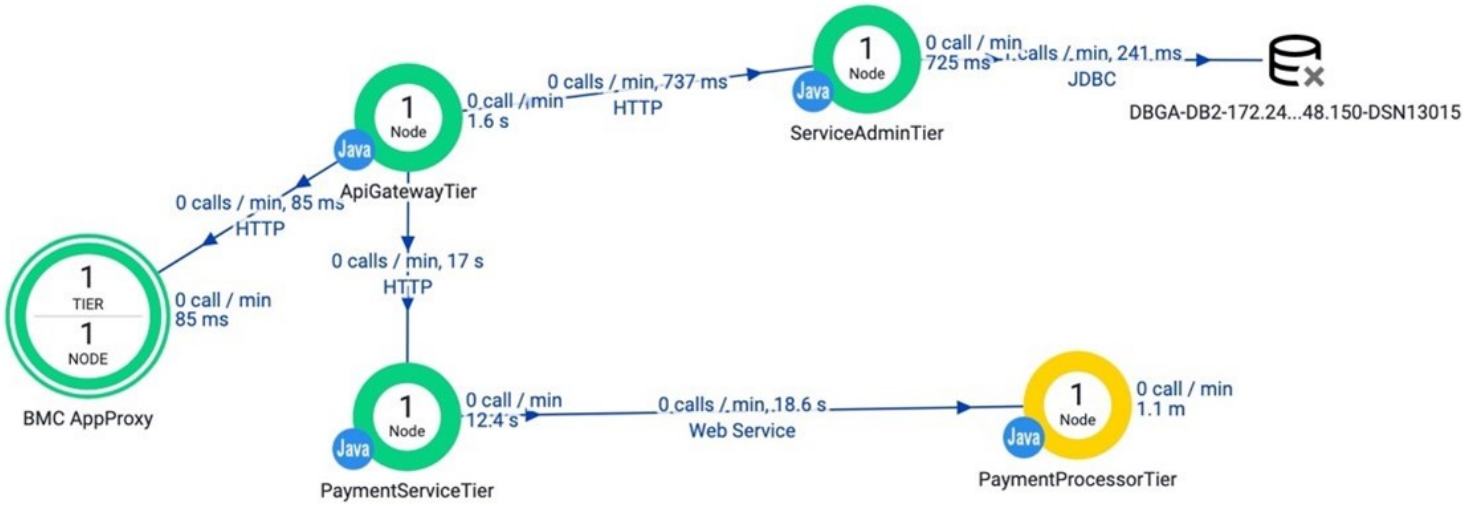

Figure 5. AppDynamics topology (limited to application components).

The blueprint and graph database will need to do the hard work to reconcile topology across the many layers (application, infrastructure, and network).

When building a dynamic service, the service modeler will just select the software cluster (which is an application name in the AppDynamics model). The blueprint will then automatically model a service that connects the application topology to physical hosts and network devices discovered using a discovery tool and further enriched by other monitoring tools. In seconds, the service modeler can automatically create a service model—that spans application to network and cloud to mainframe—for AIOps, change management, and asset management use cases. This service model is dynamic and will be automatically maintained based on the blueprint. If, in the future, CIs are added or removed, the pattern matching will dynamically account for any changes and maintain a current service model.

Figure 6 shows the result of using the AppDynamics blueprint. You can see the relationships between software components, runtime processes, virtual hosts, physical hosts, and interface cards/port connected to a switch. This level of detail is required for change impact analysis and AI/ML algorithms to perform root cause analysis when a critical business service is at risk or at fault.

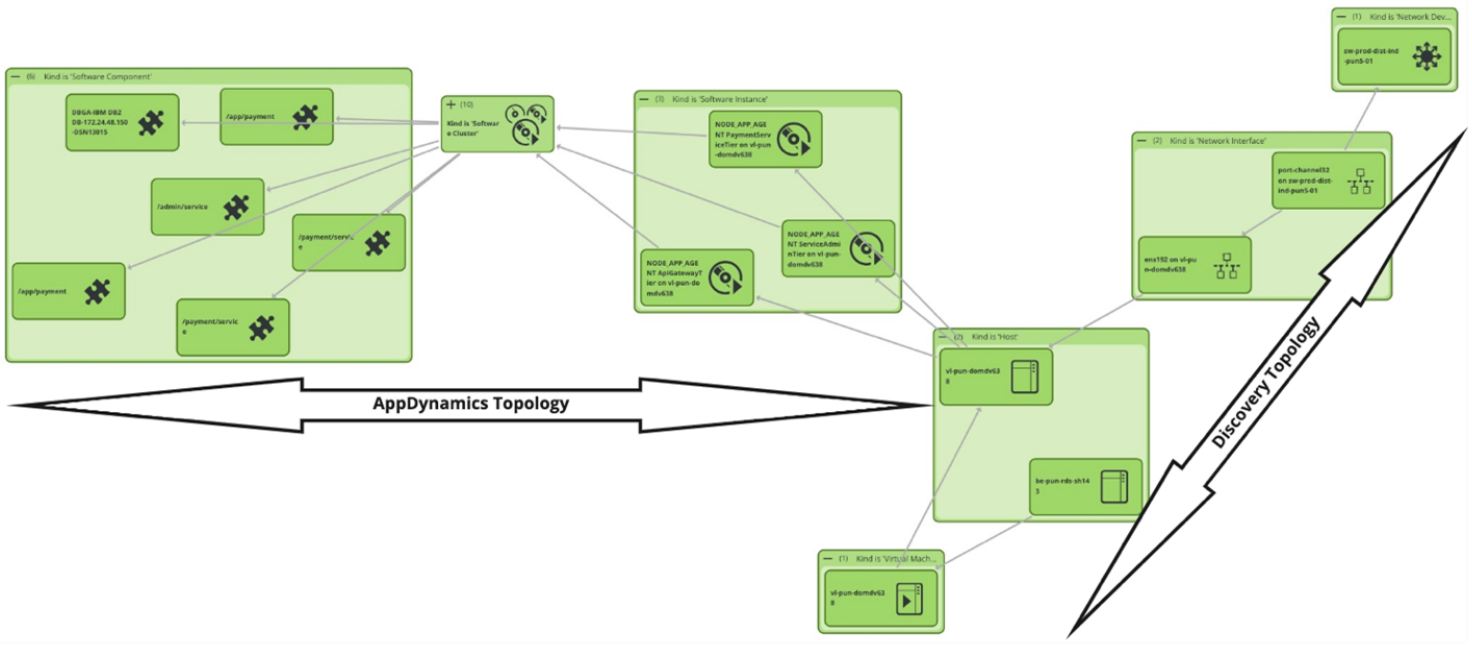

Figure 6. AppDynamics blueprint result (application to network topology).

Show inter-service impacts by modeling service dependencies

Blueprints for infrastructures or core networks can also be used to model low-level virtual or physical shared services, and customer-facing services can be modeled to depend on these shared services. A service hierarchy map helps provide answers on which services, for example, are impacted by a shared service like core network, infrastructure, or storage. Figure 7 below shows how a core network issue impacts a customer-facing retail outlet service.

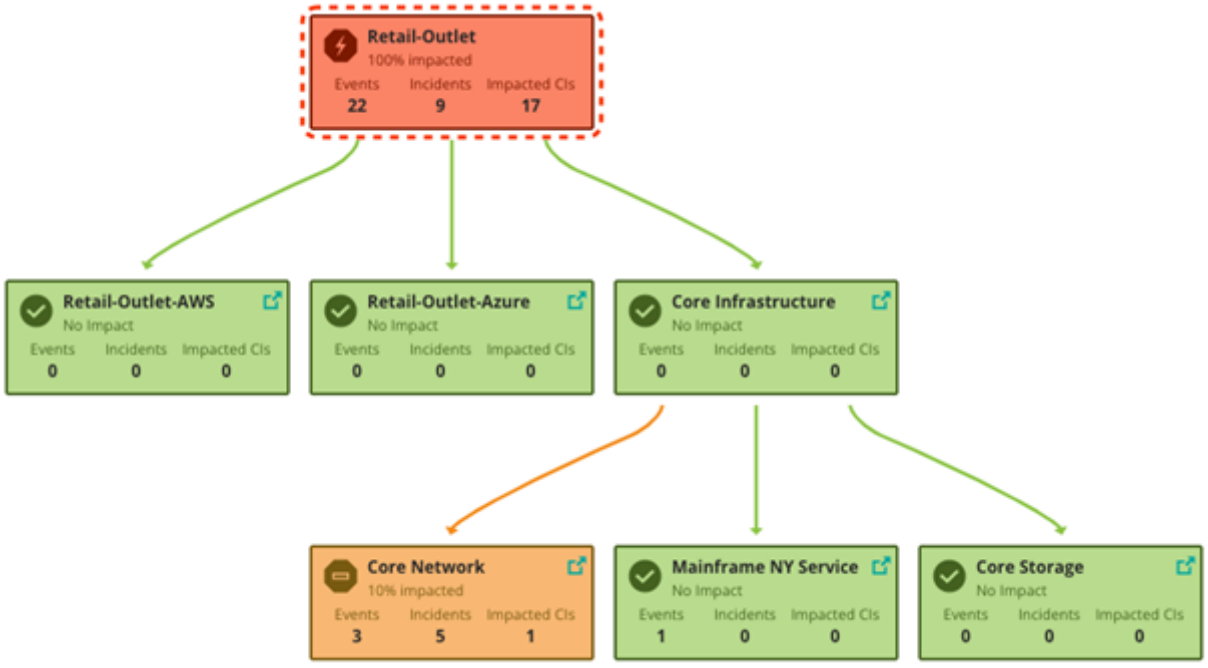

Figure 7. Shared service impact on retail outlet application.

In a future blog post, I will show how we can easily enhance the AppDynamics blueprint to build an uber-blueprint that includes multi-cloud, mainframe, and storage for a distributed microservices application.