StackState

The Observability Maturity Model

Table of Contents

Preface

At StackState, we have spent eight years in the monitoring and observability space. During this time, we have spoken with countless DevOps engineers, architects, SREs, heads of IT operations and CTOs, and we have heard the same struggles over and over.

Today's consumers are used to great technology that works all the time. They have little tolerance for outages or performance issues. These expectations push businesses to stay competitive through frequent releases, ever-faster response and greater reliability. At the same time, the move towards cloud-based applications – with all of their ever-changing functions, microservices and containers – makes IT environments more complex and harder than ever to operate and monitor.

As a result, we have seen great commonalities in the monitoring challenges that are unfolding globally, such as this colorful issue described by a customer:

> "When something big broke in the infrastructure, storage, networking equipment or something like that... every time we saw the same movie. The monitoring gets red, red, red, thousands of alarms, nobody knows what's the root cause. Everybody is panicked – real total chaos."
>
> - Georg Höllebauer, Enterprise Metrics Architect at APA-Tech

I witnessed this problem first-hand eight years ago when I was part of a team of two consultants working at a major Dutch bank, helping them improve the reliability of their mission-critical applications. They were a mature enterprise with multiple monitoring tools in place for their complex environment, but they could not find the root cause of problems quickly. As a result of many siloed tools and lack of unified view of their IT environment, customer experience was directly suffering. When something broke, it took too long to find and fix the core problem. We knew we had to find a better way, and the technology we built to meet this bank's needs became the foundation for StackState.

Since we released the original Monitoring Maturity Model in 2017, it has become clear that the original monitoring tools – which simply notified IT teams when something broke – were no longer sufficient for many other organizations as well. Today's engineers need to immediately understand the priorities and context surrounding a problem: what's the impact on customer experience and business results? Then, if the impact is high: why did it break and how do we fix it?

The concept of observability has evolved from monitoring to answer those questions. Observability is vital in maintaining the level of service reliability needed for business success. Unfortunately, navigating the monitoring and observability space is hard, especially as AIOps enters the picture. Many vendors are making a lot of noise in the market and new open source projects are popping up left and right. It's hard to know who really does what, and even harder to know which capabilities really matter.

The Observability Maturity Model is based on extensive experience with real problems in live environments, discussions with customers and prospects, research into the latest technologies and conversations with leading analyst firms such as Gartner. We hope it will help you shine some light in the darkness. Our goal is not to present you with the perfect model of what your observability journey should look like. We know it doesn't work like that. To quote a famous British statistician, "All models are wrong, some are useful." Rather, we wrote this Observability Maturity Model to help you identify where you are on the observability path, understand the road ahead and provide a map to help you find your way.

May this model be useful to you on your journey!

Lodewijk Bogaards
Co-founder and Chief Technology Officer
StackState

Introduction: Why the Observability Maturity Model?

Monitoring has been around for decades as a way for IT operations teams to gain insight into the availability and performance of their systems. To meet market demands, innovate faster and better support business objectives, IT organizations require a deeper and more precise understanding of what is happening across their technology environments. Getting this insight is not easy, as today's infrastructure and applications span multiple technologies, use multiple architectures and are more dynamic, distributed and modular in nature.

Change is also a way of life in IT and research shows 76% of problems are caused by changes.[1] In order to maintain reliability in the face of all these challenges, a company's monitoring strategy must evolve to observability.

> 66% of MTTR is spent on identifying change that is causing a problem.[3]

Most enterprises find it difficult to find the right monitoring strategy to manage their environments reliably. Over 65% of enterprise organizations have more than 10 monitoring tools, often running as siloed solutions.[2] This segregated structure limits the ability of SRE and IT operations teams to detect, diagnose and address performance issues quickly. When issues occur, teams try to find the root cause by combining teams, processes and tools, or by manually piecing together siloed data fragments. This traditional approach to monitoring is time consuming and does not provide the insights needed to improve business outcomes. Troubleshooting is just too slow and your most crucial customer-facing systems may be down for hours, resulting in millions in lost revenue.

The move to dynamic cloud, containers, microservices and serverless architectures, combined with the need to maintain hybrid environments and legacy systems of record, further exacerbates the need for more advanced capabilities.

Observability practices have evolved to meet these needs, combining advances in monitoring with a more holistic approach that provides deeper insights and a more precise understanding of what is happening across technology environments. The Observability Maturity Model defines four distinct levels in the evolution of observability, as described in Table 1 on the following page.

> Cloud and container migrations are driving the need for greater observability maturity.

Level	Goal	Functionality
1. Monitoring	Ensure that individual components are working as expected.	• Tracks basic health of individual components in IT systems • Looks at events; triggers alerts and notifications • Tells you *that* something went wrong... but not *what*
2. Observability	Determine why the system is not working.	• Gives insights into system behavior by observing its outputs • Focuses on results inferred from metrics, logs and traces, combined with existing monitoring data • Delivers baseline data to help investigate what went wrong and why
3. Casual Observability	Find the cause of the incident and determine its impact across the system.	• Provides more comprehensive insights to help determine what *caused* a problem • Adds ability to track topology changes in the IT stack over time, building on Level 1 and Level 2 foundations • Generates extensive, correlated information that helps reduce time needed to identify what went wrong, why the issue occurred, when it started and what other areas were impacted
4. Proactive Observability With AIOps	Analyze large volumes of data, automate responses and prevent anomalies from becoming problems.	• Uses AI and ML to find patterns in large volumes of data • Combines AI/ML with data from Levels 1-3 to provide the most comprehensive analysis across the stack • Detects anomalies early and gives sufficient warnings to prevent failures

Table 1: Defining the levels of observability maturity

Each level of observability builds on the foundation established in previous levels to add capabilities in capturing, tracking and analyzing data. The new functionality enables deeper observability at each stage, resulting in improved IT reliability and customer satisfaction, as shown in Figure 1 below. Although you can marginally improve results within a level by enhancing processes, most teams need to collect new types of data to advance to the next maturity level and realize greater benefits.

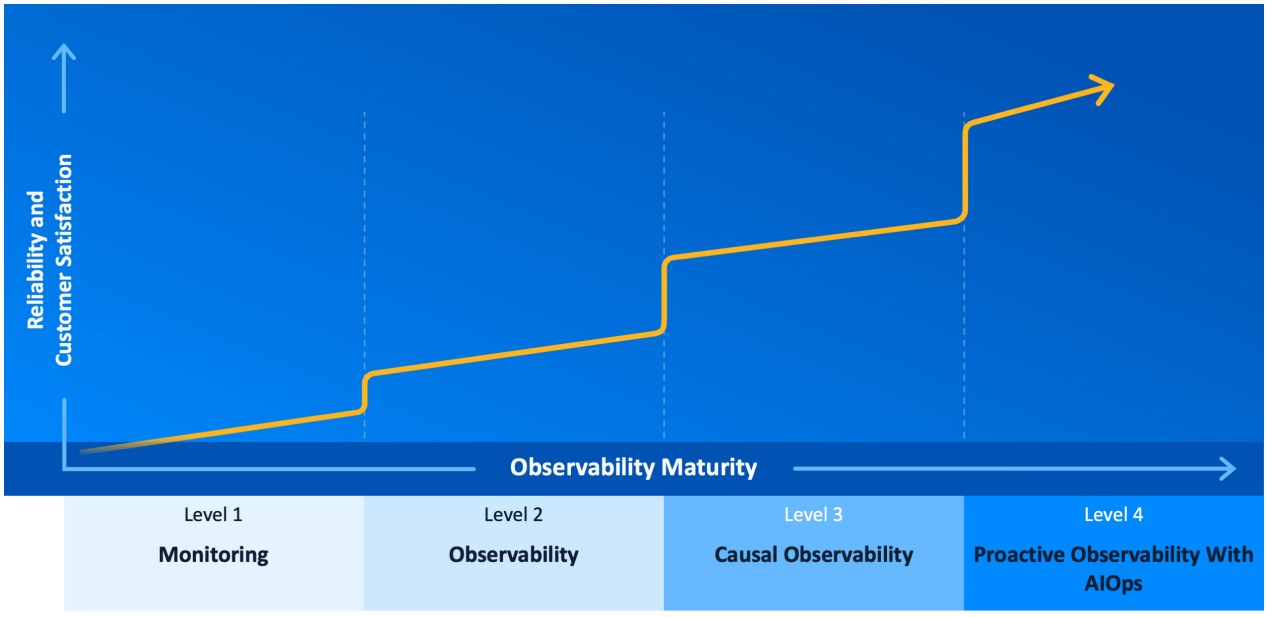

Figure 1: Observability maturity and how it affects IT reliability

The Observability Maturity Model is based on research and conversations with enterprises across industries and has been validated with other practitioners, analysts and thought leaders. It is designed to help you:

- Understand different types of data and how monitoring and observability practices can help your organization collect actionable information.
- Understand the differences between monitoring, observability and AIOps.
- Evaluate your organization's current level of maturity.
- Guide your team to a higher level of maturity.

Use this model to learn clear steps you can take to improve observability in your organization so you can ultimately deliver more reliable and resilient applications to your customers.

Level 1: Monitoring

Goal: Ensure that individual components are working as expected.

The first level, Monitoring, is not new to IT. A monitor tracks a specific parameter of an individual system component to make sure it stays within an acceptable range. If the value moves out of the range, the monitor triggers an action, such as an alert, state change, notification or warning.

With traditional monitoring, which often encompasses application performance monitoring (APM), infrastructure monitoring, API monitoring, network monitoring and various other domain-centric tooling, the use case is, "Notify me when something is not operating satisfactorily." You can think of monitoring in terms of traffic light colors:

- The component is available and healthy (green)
- The component is at risk (orange or yellow)
- The component is broken (red)

Monitoring looks at pre-defined sets of values with pre-defined sets of failure modes. It focuses on basic component-level parameters, such as availability, performance and capacity and generates events that report on the state of the monitored value.

Events are noteworthy changes in the IT environment. Though events may be purely informative, they often describe critical incidents that require action. Events may trigger **alerts or notifications** that arrive via various channels, such as email, chat, a mobile app or an incident management system.

As a first step towards observability, implement monitoring to get basic insights into the health and status of individual components and be notified when something breaks. Below, Table 2 gives an overview of the key capabilities for Level 1.

Level 1: Monitoring	
Use basic traffic-light monitoring to understand the availability of the individual components that make up your IT services.	
System Input Events and component-level metrics (e.g., "API response time is higher than our SLO of five seconds")	**System Output** Alerts or notifications (e.g., "order fulfillment service is down")
What You Get • Basic information such as the health status of a component — is it working? • Alerts and notifications when issues occur • Easiest way to get started; many open-source and SaaS solutions are available	

Table 2: Level 1 summary

Next Step: Observability

Monitoring gives you limited insights into the state of the overall environment. It shows you individual component health but generally no information about the big picture. It tells you something is broken but not why, who to call, nor when and where the original problem started.

Setting up and maintaining monitoring checks and notification channels requires a lot of manual work. At Level 1, you also need to do root cause analysis and impact analysis manually and you have a limited set of data. Investigating the sources of problems takes time. In addition, a single issue may cause storms of alerts from multiple components, causing further confusion and delays in pinpointing the root cause.

While monitoring can detect a limited number of known types of failures, or "known unknowns," Level 2, Observability, can help you discover unknown and unexpected failure modes, or "unknown unknowns." As you move from Level 1 to Level 2, you will gain more in-depth information that provides a better understanding of the availability, performance and behavior of your services.

Level 2: Observability

Goal: Determine why the system is not working.

To keep today's complex and dynamic IT systems running reliably, you need to not only know what's working (monitoring) but also understand why it's not working (observability).

Traditional monitoring tracks the basic health of a component or system. Observability evolved naturally to provide *deeper insights into the behavior of* a system over time. When something goes wrong and your team receives an alert, you need to quickly figure out, "What happened? Where, when, why and who do we call?" Observability data helps you answer these questions. At its full maturity (Level 4), observability provides all the data you need, in the proper context, to automatically detect and remediate issues and even to proactively identify and prevent them.

When an alert pops up, you look to understand the state of your system to find the problem's source. At Level 2, observability typically delivers system insights by focusing on three critical types of telemetry data: **metrics**, **logs** and **traces**.[4] These *three pillars of observability* are collected from IT components such as microservices, applications and databases to provide an overall perspective into a system's behavior. Each pillar gives a different type of information, as outlined in Table 3 below.

Pillar	Definition
Metrics	Numerical measurements that help you understand the performance and status of services — for example, the famous four golden signals: latency, traffic, error rate and saturation.[5]
Logs	Time-stamped records of relevant events that happen in a system (e.g., transactions, warnings, errors), which help you understand a system's behavior at a given point in time.
Traces	Detailed snapshots showing how data flows through an application from end to end (e.g., a user request), which help troubleshoot performance and sometimes give code-level visibility into how your app performs.

Table 3: Three pillars of observability

These three pillars, along with events and alerts, are typically plotted on dashboards so teams can easily keep track of important activities. Some observability tools provide out-of-the box dashboards that bring together these different types of data on one screen and allow you to deep-dive into them for further investigation.

Level 2 data has much greater breadth and depth than Level 1, and it often involves some data consolidation across your environment into a single view. You may need to build additional dashboards if you want more insights, especially if your environment has multiple domains and you are using multiple monitoring tools.

Level 2: Observability	
Observe the behavior of IT environments by capturing metrics, logs and traces in addition to events and health state.	
System Input Level 1 inputs + comprehensive metrics, logs and traces	**System Output** Level 1 outputs + comprehensive dashboards with graphs, gauges, flame charts, logs, etc.
What You Get • Deeper, broader and more holistic view of overall system health by collecting additional data from more sources, which better supports problem diagnosis • Ability to discover unknown failure modes in addition to known types of failures • Beneficial insights from individual types of data — e.g., traces help identify performance bottlenecks, metrics make excellent KPIs and logs can be used to find software defects	

Table 4: Level 2 summary

The challenge then becomes how to resolve information from too many dashboards. At Level 2, you can infer suspected reasons for incidents by manually correlating data, but this approach often involves complex manual queries across systems.

At Level 2, teams have not yet developed an automated way to unify and correlate the siloed data from various tools and domains, so it is still labor intensive and time consuming to pinpoint the root cause of an issue. Consequently, MTTD and MTTR are higher than they should be, customers are more adversely affected and more revenue is lost than at higher maturity levels.

Next Step: Causal Observability

Observability generates a huge amount of data and sorting out the meaningful information can be difficult.

At Level 2, your team is likely challenged by both data silos and volume, which cause inefficiencies in cross-domain and cross-team troubleshooting.

When something goes wrong, too many people get involved because nobody knows where the problem is, resulting in incident ping-pong and blame games. You may need to build ad hoc solutions to query multiple observability silos to troubleshoot a single issue. Creating these queries requires practitioners with development skills, knowledge of data structures and understanding of system architecture.

In addition, the telemetry-centric and siloed views typical in Level 2 often require substantial manual work to extract actionable insights. Setting up efficient dashboards can take considerable time and

they require ongoing maintenance. Root cause analysis, impact analysis and alert noise reduction are important in maintaining a reliable and resilient stack, but these activities are challenging at this level.

Note: Teams are increasingly adopting the OpenTelemetry standard to facilitate the capture of metrics, logs and traces. OpenTelemetry is extremely helpful to efficiently collect these types of data, but it was not designed to bridge silos, create better context for data or to analyze the data.

In order to move to Level 3 and understand how your observability data is related, you need to provide context for events, logs, metrics and traces across the data silos in your IT environment. At Level 3, Causal Observability, you get a precise map of the topology of your business processes, applications and infrastructure and you can track how it all changes over time. When something goes wrong, you can use this contextual data combined with automation to quickly determine the cause of an issue without having to manually wade through silos of uncorrelated data.

Level 3: Causal Observability

Goal: Find the cause of the incident and determine its impact across the system.

It's not surprising that most failures are caused by a change somewhere in a system, such as a new code deployment, configuration change, auto-scaling activity or auto-healing event. As you investigate the root cause of an incident, the best place to start is to find what changed.

To understand what change *caused* a problem and what *effects* propagated across your stack, you need to be able to see how the relationships between stack components have changed over time:

- What did the stack look like when a problem began?
- What components are affected?
- How are all the alerts related?

We call this level of insight, which lets you track *cause* and *effect* across your stack, *causal observability* — it builds on the foundation laid in Levels 1 and 2.

> **"Deriving patterns from data within a topology will establish relevancy and illustrate hidden dependencies. Using topology as part of causality determination can greatly increase its accuracy and effectiveness."**
>
> – Gartner® Market Guide for AIOps Platforms, May 2022, Pankaj Prasad, Padraig Byrne, Gregg Siegfried

Topology is the first necessary dimension for causal observability. Topology is a map of all the components in your IT environment that spans all layers, from network to application to storage, showing how everything is related. Topology incorporates logical dependencies, physical proximity and other relationships between components to provide human-readable visualization and operationalized relationship data.

> **Topology describes the set of relationships and dependencies between the discrete components in an environment, for example, business services, microservices, load balancers, containers and databases.**
>
> **In today's modern environments, topologies evolve quickly as new code gets pushed into production continuously and the underlying infrastructure changes rapidly. Managing these dynamic environments requires the ability to track changes in topology over time (time-series topology), giving historical and real-time context to the activities happening in your stack.**

Modern environments consist of so many dynamic layers, microservices, serverless applications and network technology that adding an up-to-date topology to your observability mix is essential to separate cause from effect. Topology provides anchor points for thousands of unconnected data streams to give them structure, making previously invisible connections visible. Topology visualization lets you view telemetry from network, infrastructure, application and other areas in the context of full-stack activity; it also gives you crucial context to know how your business is affected when something breaks.

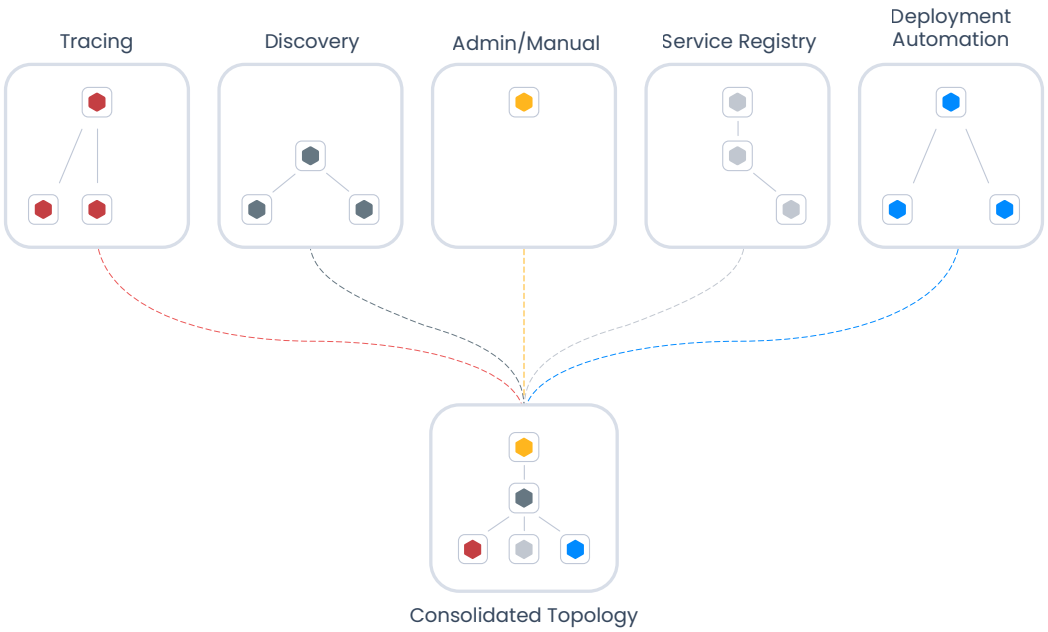

Figure 2: Causal observability requires the consolidation of topology information from all the sources in your environment.

However, for most companies, adding topology is not enough to provide causal observability on its own. Especially in today's dynamic modern environments with microservices, frequent deployments, ever-changing cloud resources and containers spinning up and down, topology changes fast. What your stack looks like now is probably not what it looked like when a problem first began. So a second dimension is necessary to create the foundation for causal observability: time.

Figure 3: Capture time-series topology to track stack changes and quickly troubleshoot root cause.

And finally, to understand the dynamic behaviors of modern IT environments and get the context required to achieve causal observability, you need to *correlate* your environment's *topology* with its associated *metric*, *log*, *event* and *trace data* over time.

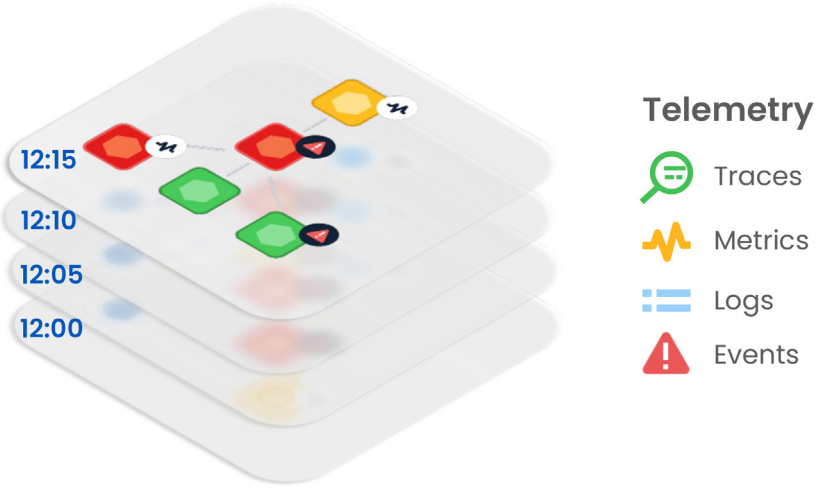

Figure 4: Capture topology over time and correlate it with metrics, logs, events and traces to track changes in your stack. Later, when issues occur, you can go back to the exact moment in time the issue started and see what change caused it.

At Level 3, the additional dimensions of topology and time, correlated with telemetry data show you the cause and impact of any change or failure across the different layers, data silos, teams and technologies — significantly improving resolution times and business outcomes. You also have the foundation to begin automating root cause analysis, business impact analysis and alert correlation. This deeper level of data is also required for more advanced AIOps, as you'll read about in Level 4.

4 Key Steps to Build Causal Observability and a Foundation for AIOps

1. **Consolidate:** First, you need to ensure you have **consolidated data** from across your stack into one place so you have a complete view.
2. **Collect topology data:** Next, you need to **build a topology map** of your environment, which is a map of the components in your stack showing how they all relate to each other. Visualizing topology quickly answers the questions, "What component depends on other components? If one service fails, what else will be affected?"
3. **Correlate:** You need to **correlate** all this unified data so your entire IT environment can be analyzed as a whole, even across silos. Every component in the topology needs to be correlated with its associated **metric**, **log**, **event** and **trace** data.
4. **Track everything over time:** Finally, if you want to see how a change in one component propagates across your stack, you need to correlate your topology data with metric, log and trace data **over time**.

Level 3: Causal Observability	
Contextualize telemetry data (metrics, traces, events, logs) through a single topology. Correlate all data over time to track changes as they propagate across your stack.	
System Input Levels 1 and 2 + time-series topology	**System Output** Levels 1 and 2 + correlated topology, telemetry and time data displayed in contextual visualizations, showing the effects of changes across your stack
What You Get • Consolidated, clear, correlated, contextual view of the environment's state, through unification of siloed data in a time-series topology • Significant acceleration in root cause identification and resolution times through topology visualization and analysis to understand cause and effect • Foundation for basic automated investigations such as root cause analysis, business impact analysis and alert correlation • Context needed to automatically cluster alerts related to the same root cause, reducing noise and distractions • Ability to visualize the impact of network, infrastructure and application events on business services and customers	

Table 5: Level 3 summary

Next Step: Proactive Observability With AIOps

As noted above, Gartner points out that topology can greatly increase the accuracy and effectiveness of causal determination. Level 3 is a big step forward, but unifying data from different silos poses challenges in terms of data normalization, correlation and quality that may require new capabilities or even organizational changes to resolve. In addition, it is difficult to collect and operationalize high-quality topology data at scale, especially in less modern environments.

Each topology source needs to continuously flow through into the master topology, so you need to ensure you have a system with the capability to store topology over time. Storing topology that is correlated with telemetry data over time presents an even bigger challenge.

Consider these issues as you develop your implementation plan. Also keep in mind that the velocity, volume and variety of data at Level 3 is usually so large that to achieve your overall reliability goals, AI is likely necessary to help separate the signal from the noise. When you take the step to Level 4, you add artificial intelligence for IT operations (AIOps) on top of Levels 1-3 to gain more accurate insights.

> **"With data volumes reaching or exceeding gigabytes per minute across a dozen or more different domains, it is no longer possible, much less practical, for a human to analyze the data manually in service of operational expectations."**
>
> – Gartner® Market Guide for AIOps Platforms, May 2022, Pankaj Prasad, Padraig Byrne, Gregg Siegfried

Level 4: Proactive Observability With AIOps

Goal: Analyze large volumes of data, automate responses to incidents and prevent anomalies from becoming problems.

Level 4, Proactive Observability With AIOps, is the most advanced level of observability. At this stage, artificial intelligence for IT operations (AIOps) is added to the mix. AIOps, in the context of monitoring and observability, is about applying AI and machine learning (ML) to sort through mountains of data looking for patterns that

- drive better responses
- at the soonest opportunity
- by both humans and automated systems.

In Gartner's "Market Guide for AIOps Platforms," May 2022, by Pankaj Prasad, Padraig Byrne and Gregg Siegried, Gartner defines the characteristics of AIOps platforms in the following way:

> "AIOps platforms analyze telemetry and events, and identify meaningful patterns that provide insights to support proactive responses. AIOps platforms have five characteristics:
>
> 1. Cross-domain data ingestion and analytics
> 2. Topology assembly from implicit and explicit sources of asset relationship and dependency
> 3. Correlation between related or redundant events associated with an incident
> 4. Pattern recognition to detect incidents, their leading indicators or probable root cause
> 5. Association of probable remediation"

We have the same view on AIOps as Gartner. AIOps builds on core capabilities from previous levels in this maturity model — such as gathering and operationalizing data, topology assembly and correlation of data — and adds in pattern recognition, anomaly detection and more accurate suggestions for remediating issues. Causal observability is a necessary foundation: time-series topology provides an essential framework.

AIOps can help teams find problems faster and even prevent problems altogether. AI/ML algorithms look for changes in patterns that precede warnings, alerts and failures, helping teams know when a service or component starts to deviate from normal behavior and address the issue before something fails.

> "Spotting an anomaly is easy because they occur all the time. When you collect one billion events a day, a one-in-a-million incident happens every two minutes. The key for observability tools is to spot the anomaly that is relevant to the problem at hand, and then to link other bits of information from log files / metrics that are likely to be related. By surfacing correlated information in context, the operator can more quickly isolate the potential root cause of problems."
>
> – Gartner® "Innovation Insight for Observability," March 2022, Padraig Byrne and Josh Chessman

However, anomalies occur frequently. They do not necessarily mean a problem will occur, nor that remediation should be a high priority. AIOps helps determine which anomalies require attention and which can be ignored.

Another goal of AIOps for observability is to drive automated remediation through IT service management (ITSM) and self-healing systems. If these systems receive incorrect root cause input, for example, they can self-correct the wrong issue and cause bigger problems. AIOps delivers more accurate input that enhances their effectiveness.

> An ounce of prevention is worth a pound of cure. What better way to improve reliability than to stop incidents from ever happening at all?

At Level 4, you should notice more efficient and incident-free IT operations that deliver a better customer experience. To achieve these goals, set up AIOps to transcend silos and ingest data gathered from across the environment. The AI/ML models should analyze all the observability data types we discussed in previous levels: events, metrics, logs, traces, changes and topology, all correlated over time.

A Word of Caution: Don't Skip Level 3

Proactive observability with AIOps is the best way to ensure reliable operation of your IT systems, but it's a mistake to move directly to Level 4 and skip over the causal observability steps in Level 3 (data consolidation, topology, correlation of all data streams over time).

Each level in this Observability Maturity Model builds on capabilities established in previous levels, but having a complete foundation matters most for success in Level 4. If you apply AI/ML without a comprehensive foundation of data, you can actually cause damage. For example, let's say you use AI/ML on the front end of an automated self-healing system. If the algorithm determines an incorrect root cause, the self-healing system tries to remediate the wrong thing and can further break the system. If you apply AI/ML on top of insufficient data or poor-quality data, you may drive automation in the wrong direction as the algorithm learns the wrong thing.

Without topology data correlated with metric, log and trace data over time, AIOps tools will likely not understand the correlation between these various sorts of data as they come together. AIOps needs the additional context provided by topology and time in order to accurately assess root cause, determine business impact, detect anomalies and proactively determine when to alert SRE and DevOps teams.

Level 4: Proactive Observability With AIOps	
Use AIOps to sort through mountains of data and identify the most significant patterns and impactful events, so teams can focus their time on what matters.	
System Input Levels 1-3 + AI/ML models	**System Output** Levels 1-3 + proactive insights that enable fast MTTR and prevent failures
What You Get • New insights into IT environment operations using AI/ML to gather and correlate actionable information from large volumes of data • Predictions and anomaly detection that highlight issues before they impact the business • Greater efficiency and reduced toil as teams focus effort on the most impactful events • Improved accuracy of automatic root cause analysis, business impact analysis and alert correlation • Incident data that is accurate enough to use effectively with automated ITSM and self-healing systems	

Table 6: Level 4 summary

Next Steps

Most AIOps solutions today require significant configuration and training time but often yield inaccurate results, especially if topology changes over time are not considered. Teams often implement them with unrealistic expectations and unclear goals, then find themselves disappointed.

Level 4 is the final observability maturity level for now, but as IT continues to evolve, we fully expect a Level 5 to emerge.

Summary

For decades, IT operations teams have relied on monitoring for insight into the availability and performance of their systems. But the shift to more advanced IT technologies and practices is driving the need for more than monitoring – and so observability evolved. With infrastructures and applications that span multiple dynamic, distributed and modular IT environments, organizations need a deeper, more precise understanding of everything that happens within these systems. Observability provides that comprehensive insight, delivering clear capabilities at each level of maturity.

Drivers to Improve Maturity	
Level	**Drivers**
Level 1: Monitoring	Level 1 is sufficient for classic static infrastructure.
Level 2: Observability	Level 2 capabilities become more critical as you shift to cloud, container and microservices architectures and implement CI/CD.
Level 3: Causal Observability	Level 3 capabilities become essential for maintaining hybrid environments, expanding to multi-cloud platforms, implementing containers, microservices and more advanced CI/CD at scale.
Level 4: Proactive Observability with AIOps	As companies attempt to automate systems for event correlation, automatic ticket creation, ticket consolidation, automatic remediation and self-healing, Level 4 capabilities for AIOps are required. The intelligence provided by AIOps delivers the data accuracy necessary for these systems.

Figure 5: Typical technology environments that drive companies to advance their observability maturity.

Each level of observability is characterized by distinct goals, inputs, outputs and capabilities. You'll also find commonalities in typical tooling at each level.

	Level 1: Monitoring	Level 2: Observability	Level 3: Causal Observability	Level 4: Proactive Observability With AIOps
Observability Goals				
Ensure that individual components are working as expected.	●	●	●	●
Determine why the system is not working.		●	●	●
Find the cause of the incident and determine its impact across the system.			●	●
Analyze large volumes of data, automate responses and prevent anomalies from becoming problems.				●
System Input				
Events and component-level metrics	●	●	●	●
Metrics, logs, traces (comprehensive)		●	●	●
Time-series topology			●	●
AI and ML models				●

	Level 1: Monitoring	Level 2: Observability	Level 3: Causal Observability	Level 4: Proactive Observability with AIOps
System Output				
Alerts	●	●	●	●
Comprehensive dashboards		●	●	●
Understand cause and effect of change			●	●
Automated root cause analysis			●	●
Automated business impact analysis			●	●
Correlated alerts / noise reduction			●	●
Predictive and preventative insights				●
Typical Tooling				
Classic domain-centric monitoring tools (e.g., infrastructure monitoring, application monitoring, API monitoring, synthetic monitoring, network monitoring, business monitoring), event-based alerting.	●	●	●	●
APM/observability tooling – APM tools, modern observability tools based on OpenTelemetry, observability data lakes (previously known as log aggregators), domain-agnostic combinations of open source metrics, trace and log tooling, sometimes unified in dashboard tooling.		●	●	●
More advanced APM and observability tooling with causal reasoning and event correlation capabilities, powered by time-series topology. Level 3 is an emerging market area.			●	●
Data-agnostic AIOps solutions that can find patterns in large amounts of data to provide smart capabilities, such as anomaly detection and leading indicator detection/proactive alerting. Level 4 is an emerging market area.				●

Figure 6: Characteristics at each level of observability maturity. Where does your organization best fit?

The higher your maturity level, the more resilient and reliable your IT systems will be. You'll be able to troubleshoot the root cause of problems more quickly, understand business impact of changes and failures and ultimately deliver a better experience for customers.

References

1 "18 Key Areas Shaping IT Performance Markets in 2020," Digital Enterprise Journal (DEJ)

2 Enterprise Management Associates (EMA) APM Tools Survey

3 "2022 State of Managing IT Performance Study – Key Takeaways," Digital Enterprise Journal (DEJ)

4 *Distributed Systems Observability: A Guide to Building Robust Systems: A Guide to Building Robust Systems*, by Cindy Sridarhan, O'Reilly Media, 2018.

5 *Site Reliability Engineering: How Google Runs Production Systems*, edited by Betsy Beyer, Chris Jones, Jennifer Petoff and Niall Richard Murphy, O'Reilly Media, 2016.

About StackState

StackState delivers a new and essential capability for managing today's complex hybrid, cloud and container environments. Our unique approach, which we call topology-powered observability, captures the complete topology of your stack and correlates this data with telemetry and trace information at every moment in time. Only StackState has versioned graph database technology that can store every topology change in your stack at scale, enabling you to track dynamic relationships that help identify root cause. This foundation provides a complete picture of the state of your stack and lets you apply the intelligence you need to quickly find, fix and prevent problems.

Leading enterprises like KPN, Vodafone, Accenture and Danske Bank rely on StackState to ensure the performance and reliability of their business-critical services. For more information, visit us at **www.stackstate.com** and follow us on Twitter, LinkedIn and Facebook.

StackState

0922v01