Observability and Maturity Levels

Lauri Humina, lauri.humina@wakaru.fi
Chief Digital Officer, Wakaru Oy

linkedin.com/in/laurihumina

Wakaru.

Understanding Observability

- Exactly "how" can the internal state of a system be known?

  With proper applications in place, forms of communication called **signals** are emitted that provide quality information to monitor the internal state of the system known as **Observability**

- Examples of Signals
  - Metrics
  - Events
  - Logs
  - Traces

- Data emitted and collected from these signals
  - Telemetry

Wakaru.

Metrics, Events, Logs, and Traces (MELT)

These are the essential data types for Observability. When we instrument everything and use **MELT** to form a fundamental working knowledge of connections—the relationships and dependencies within the system—as well as its detailed performance and health, exemplifies **Observability**.

Metrics are the values pertaining to a system/application at a certain point in time

Events are specific sequences of occurrences that take place within a system being monitored

Logs are the original data type; in their most fundamental form, logs are essentially lines of text a system or application produces when certain code blocks are executed

Traces, or more precisely, "distributed traces", are samples of causal chains of events or transactions between different components in a microservices ecosystem

Why Is Observability Important?

Observability provides insight into the functioning of a system and enables teams to make informed decisions about the system's health and performance helping to ensure reliability and availability.

Knowing the internal state of a system leads to understanding through:
• Better insight into system behaviors
• Monitoring system performances
• Helping troubleshoot problems
• Providing real-time monitoring instantly
• Increased security
• Being cost effective
• Scalability

"One of the important things that Observability enables is the ability to see how your systems behave."

Josep Prat - Open-Source Engineering Director

Wakaru.

Who are the consumers of Observability?

Operations Engineer	QA/Test Engineer	Developer	Security Engineer	Leadership
• Wants to know if systems are meeting the demands of customers • Wants to reduce signal to noise ratio in alerts • Wants to correlate data across distributed systems and get to the root cause faster • Wants to know if systems are configured with best practices • Wants to know if all systems are patched • Wants visibility into thresholds and patterns; what is normal vs an anomaly	• Wants to know how tests impact the system • Wants contextual data about bugs • Wants to know about test coverage • Wants to understand how the new features are functioning • Wants visibility into the overall health of the system as tests execute various areas of the codebase	• Wants to know how software can be optimized • Wants to understand performance impact of changes • Wants to be able to troubleshoot issues • Wants to know how the pipeline and systems are performing • Wants to be able to trace transactions through the entire system • Wants contextual information when failures occur	• Wants to know if systems are secure • Wants to know what/where software has been deployed • Wants to know patch versions • Wants to ensure compliance with corporate standards and best practices • Wants to make sure no new vulnerabilities are being introduced with applications as they are being developed • Get real-time analysis of security alerts generated by applications and network hardware	• Wants visibility into business KPIs • Wants to know about issues and risks • Wants assurances that systems are performing and costs optimized • Wants visualizations and reports that reduce signal to noise ratio

Wakaru.

Why traditional monitoring is not enough?

1. Lack of a Holistic Practice Understanding

- In the new world of distributed architecture, software delivery from multiple teams brings the challenge of multiple Observability approaches

- This commonly causes delays in finding issues when incidents occur

Wakaru.

2. Technology Silos

- Delivering software in larger enterprises often involves a mixture of technologies, not uniform cloud-only deliveries
- Connecting all components to get the big picture might be difficult

Wakaru.

3. Lack of Understanding Production

The move to the cloud is often not the only transformation that is happening. It goes hand in hand with build, test, pipeline automation, and Observability.

Teams may accelerate in deployment frequency, but without good Observability in place, they may struggle to understand the impact in production.

Wakaru.

4. Skill Retention

With so many new practices, teams might struggle covering all of them. Building robust Site Reliability Engineering might be one of the practices they struggle with.

High employee churn in teams means skill retention is challenging. Hence, Observability is key to keep the organizational knowledge.

5. Ensuring Compliance

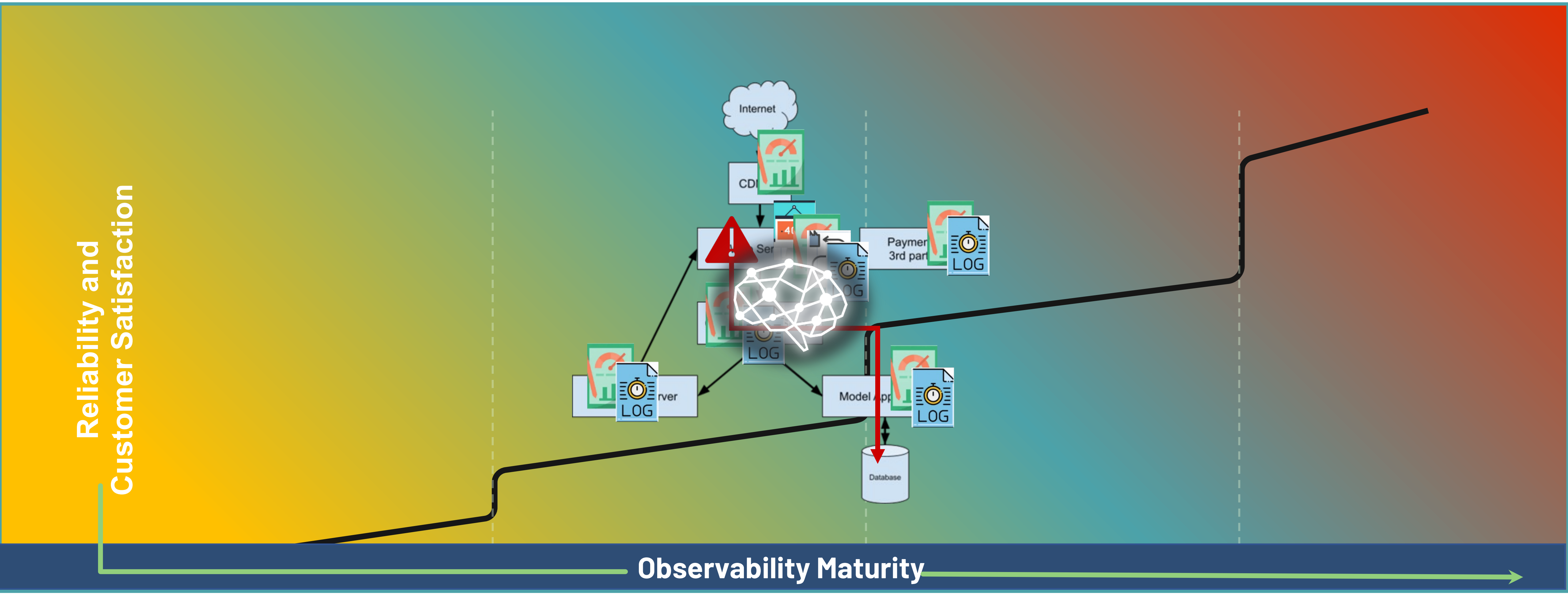
Cloud and container deployments combined with CI/CD make it much easier to deploy continuously and push out new services in minutes.

However, being compliant - knowing who does what, when - becomes more important than ever before.

Observability Maturity Model

Wakaru.

Observability Maturity Model example



Level 1	Level 2	Level 3	Level 4
Monitoring	**Observability**	**Causal Observability**	**Proactive Observability with AIOps**

Wakaru.

Why Is Maturity Model Level 1 Not Enough?

- Limited insights into the overall environment - not “big picture” so the health of the system is not known
- No longer adequate for the “always-on” society
- Setting up monitoring requires a lot of manual work
- Identifies something is broken but does not point at what
- The need to do root cause analysis and impact analysis manually
- Can detect only known types of failures

Wakaru.

Maturity Model Level 1 – Monitoring

Monitoring

Uses basic traffic light monitoring to understand the availability of the individual components that make up the IT services.

System Input

Events and component level metrics (e.g., "API response time is higher than the 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

Wakaru.

Level 2: Observability

Why is the system not working?

- This is basic Observability – an evolution of monitoring
- Begins to answer "why" on a limited scale
- Comprehensive **metrics**, **logs** and **traces**
  - "The Three Pillars of Observability"
- Necessary for dynamic on-premise, hybrid and cloud environments

Metrics

Logs

Traces

Wakaru.

Maturity Model Level 2 – Observability

Observability

Observe the behavior of IT environments by capturing metrics, logs, and traces in addition to events and the health state.

System Input

Level 1 inputs + comprehensive metrics, logs, and traces

System Output

Level 1 outputs plus comprehensive dashboards with graphs, gauges, flame charts, logs, etc.

What You Get

- Deeper, broader and more holistic view of the overall system health by collecting additional data from more sources, better supporting the diagnosis of issues
- 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

Wakaru.

Level 3: Causal Observability

There is a need to collect and correlate two additional dimensions: **Topology and Time**

Topology is a map that describes the set of relationships and dependencies between discrete components in an environment.

Time is the second essential dimension for Level 3.

- Telemetry (such as traces, metrics and logs) and topology values are needed when the problem starts, not just when it is detected

Causal Observability requires correlation of telemetry (metrics, events, logs, traces) and topology at every moment in time.

- What did the stack look like before the incident
- How did it change over time
- What components are related and affected by changes
- When a storm of alerts is received, which ones are related to the same root cause

Wakaru.

Level 3: Causal Observability

- Bringing it all together:
  Topology plus Time = Time-Series Topology
  - Shows full context and gives visibility across the IT environment including seeing business impacts when an issue occurs
  - Shows both cause and impact of change
- Could enable automated root cause analysis, impact analysis and alert correlation
- Significantly improves problem resolution times and business outcomes
- **Observability Driven Design (ODD)** enables Observability of a system throughout the entire development cycle; it's about encouraging developers to wrap their code with breadcrumbs that can be traced and increase logging and events at the coding level

Wakaru.

Maturity Model Level 3 – Causal Observability

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 plus time = series topology

System Output

Levels 1 and 2 plus 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

Level 4: Proactive Observability

How can incidents be prevented from even occurring?

- Most advanced level of Observability today
- At Level 4, AIOps is added to the mix

How does AIOps help?

- Sorts through terabytes of data generated by Observability tools
- Applies AI and ML to learn patterns that drive early and accurate responses
    - Proactive Observability – detects the anomalies *that matter*
    - Automated remediation

Maturity Model Level 4 – Proactive Observability

Proactive Observability

Uses 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 efforts 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

Wakaru.

Challenges with Observability Maturity Levels

Is there a need for Open Observability Maturity Model?

- Current models are mostly vendor specific, focus on functions in the tools

Should we focus more on outcomes?

- Unrivaled digital experience
- Flawless and secure digital interactions
- Speed-up innovation
- Optimize operational cost
- Reduce cloud complexity

How do we measure the maturity level?

- Do we focus on the technical aspects or do we measure the outcomes
- Is the organizations maturity high if they have AIOps, but don't have a robust incident or change management process?

Should we focus more on people and process?

- Always remember People >> Process >> Technology
- Panel discussion will explore this

Wakaru.