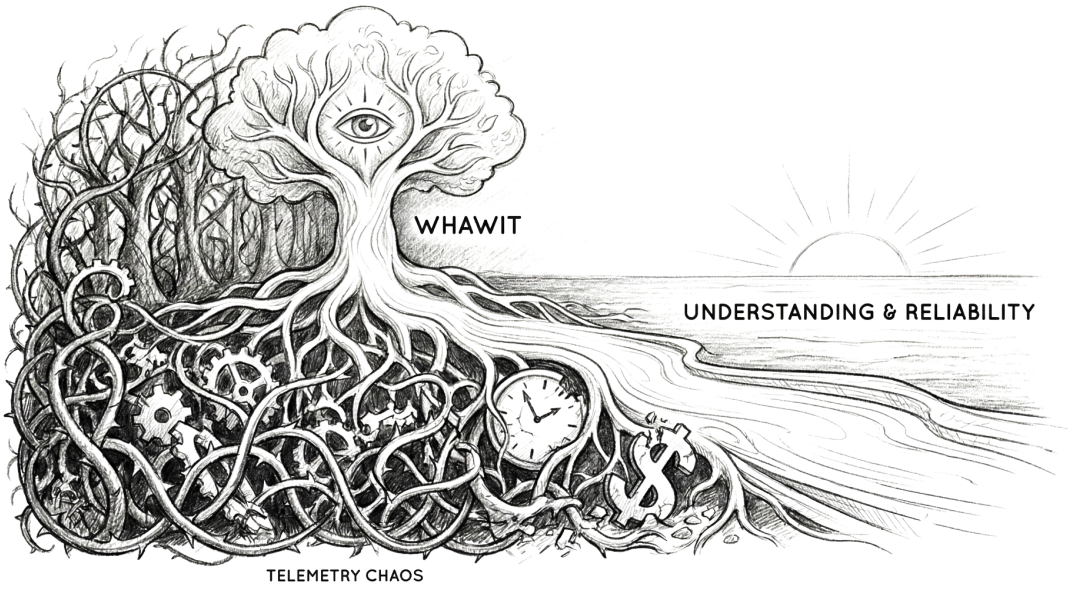
whawit

TELL ME WHAT

Intelligent Observability and Autonomous Reliability Engineering

A Research Whitepaper on MTTR,
Downtime Economics, and the Telemetry Explosion



Some illustrative images associated with the publication may have been **generated using AI-based tools** for conceptual visualization purposes only.

1. Introduction

Digital businesses now run on distributed systems: microservices, containers, serverless functions, event-driven pipelines, and third-party APIs. This architecture offers flexibility and scale, but it also introduces fragility. A seemingly small regression, a configuration mistake, or a third-party slowdown can cascade into full-blown outages.

When that happens, two questions matter more than any others:

1. **How quickly can we understand what is going on?**

2. **How quickly can we fix it—and prevent it from happening again?**

In most organizations, the answers are still unsatisfying. Teams have invested heavily in observability platforms, log pipelines, APM tools, and dashboards, yet they still rely on manual investigation and human intuition to interpret incidents under pressure. Meanwhile, telemetry volumes grow relentlessly and so do the bills for storing data that is rarely examined outside emergency situations.

This whitepaper explores the economic and operational reality behind incident response today—cost of downtime, MTTR, engineering time lost, and the burden of log storage—and then introduces WHAWIT as a **vNext layer**: an intelligent observability and autonomous reliability platform that sits on top of existing tools (Datadog, New Relic, CloudWatch, Elastic, etc.) and turns raw telemetry into understanding and improvement, not just visibility.

---

2. Market Context: The Real Cost of Downtime

Downtime has always been costly, but recent studies show just how severe the impact has become in a digital-first economy.

A 2024 study summarized by TechTarget, based on research by Oxford Economics, estimates that downtime now costs organizations an **average of 9,000 USD per minute**, or roughly **540,000 USD per hour**, with lost revenue as the primary component of direct cost.TechTarget That figure is an average across industries and company sizes; for larger enterprises it is often much higher.

New Relic's 2024 Observability Forecast reports that **45% of respondents say their high-business-impact outages cost at least 1 million USD per hour**.New Relic In heavy industrial and manufacturing contexts, the stakes are even clearer. Siemens estimates that the world's 500 largest companies collectively lose **about 1.4 trillion USD each year to unplanned downtime**, roughly 11% of their revenues.assets.new.siemens.com

The picture is similar but scaled down for smaller businesses. One 2024 analysis found that downtime can cost small organizations **around 427 USD per minute**, with some incidents leading to losses of 1 million USD per year when recurring outages and reputational damage are considered.divergeit.com ITIC's multi-year surveys have likewise shown that more than 90% of organizations, including

mid-market companies, estimate their cost of downtime at **over 300,000 USD per hour**.E-N Computers

Predictably, this financial reality has pushed observability and incident response from a purely technical domain into the boardroom. Reliability is no longer just an operational concern; it is an income statement concern.

---

3. MTTR, MTTD and Engineering Time: The Human Cost of Incidents

Because downtime is so expensive, metrics like **Mean Time To Detect (MTTD)** and **Mean Time To Recovery (MTTR)** have become central. Every minute added to MTTR multiplies the cost of an incident; every minute saved reduces both direct losses and indirect damage to reputation and customer trust.

The New Relic 2024 Observability Forecast provides a useful global reference point. Organizations that have achieved "full-stack observability" report **79% less downtime per year** than those without (70 hours versus 338 hours), and **48% lower outage costs per hour** (1.1M vs. 2.1M USD on average).New Relic+1 That is, good observability practices translate directly into fewer hours of business impact and lower cost when incidents do happen.

The human side of this equation is equally important. Every major incident consumes a significant amount of engineering time. Various studies and vendor impact analyses converge on a similar pattern: engineering teams often spend **20–30% of their time** dealing with unplanned work—incident triage, ad-hoc debugging, and navigating complex observability tooling—rather than delivering features. In some Total Economic Impact (TEI) case studies around observability platforms, consolidated monitoring and better incident tooling translated into **tens of thousands of hours per year** saved across SRE and development teams, hours that could be redeployed to higher-value work like reliability engineering and product innovation.New Relic

In other words:

- MTTR is not just a reliability metric; it is a **financial KPI**.
- Engineering time spent hunting through telemetry is a **hidden burn** that reduces the capacity of the organization to build and improve.

Reducing MTTR and the cognitive load of incidents is therefore doubly valuable: it lowers the cost of downtime and frees up engineering capacity.

---

4. The Telemetry Explosion: Logs, Metrics, and Traces at Scale

If downtime and human effort are the visible costs of failure, telemetry is the substrate on which incident response operates. Unfortunately, telemetry itself is undergoing an explosion.

Chronosphere's 2024 "State of Log Data" trends report notes that, on average, **log data grew 250% year-over-year** among surveyed organizations.Chronosphere Many teams now routinely move more than **100 GB of logs per day**, and a substantial subset generates **over 1 TB per day**.Chronosphere This growth is driven by microservices, containerization, more verbose logging practices, and a general tendency to "log first, think later."

The result is a nearly universal sense of overload. In practice:

- Log clusters become harder to scale and maintain.
- Query performance deteriorates as indices balloon.
- Costs—whether for self-hosted clusters or SaaS log platforms—rise quickly.

In theory, more logs should mean more observability and faster troubleshooting. In reality, the correlation is weak. The vast majority of data is never looked at unless an incident occurs. As one log analytics vendor succinctly put it, in many organizations **well over half of collected logs provide little day-to-day value** and sit idle until a serious outage forces engineers to dig through historical data.Chronosphere

This is the paradox at the heart of current observability practice: **data volumes grow and costs grow, but understanding does not grow at the same pace**.

---

5. The Economics of Log Storage and Observability Tooling

The financial side of this telemetry explosion is now a major pain point.

A 2023 survey commissioned by Edge Delta and conducted by Wakefield Research, covering 200 DevOps and SRE professionals, captured this sentiment directly. **Eighty-four percent** of respondents agreed that they are paying more than they should for observability, **even after limiting how much log data gets ingested**.DevOps.com+1 Ninety-three percent said their leadership is already aware of rising observability costs, and **91% expect increased scrutiny to cut those costs** within a year.DevOps.com+1

When respondents were asked about unexpected cost spikes, the most common causes were:

- Product launches and updates, which suddenly increase log and metric volumes;

- Log data mistakenly included for ingestion, such as debug logs left on in production.Edge Delta

To cope, organizations have turned to aggressive data reduction. The same research shows that **98% of companies now limit how much log data they collect** through caps, filters, or sampling.Edge Delta A large majority admits to cutting data for budget reasons rather than based on a clear value assessment. That approach has consequences: about **31% reported that they believe they have missed an outage or issue at least once because relevant data was not being collected**.Edge Delta

In parallel, TCO analyses of self-managed stacks (for example, Elasticsearch-based logging clusters) show that even seemingly modest ingest volumes can translate into **millions of dollars in total cost over a few years**, once hardware, cloud infrastructure, and personnel are included.Edge Delta Enterprises with tens of terabytes of daily ingest regularly report log infrastructure budgets that rival or exceed major application line items.

The picture is clear:

- Observability tooling and log storage have become **significant expense lines**.
- Most organizations feel they are **overpaying** relative to the value they extract.
- Data limits and truncation are being used as blunt instruments to control cost, at the risk of undermining incident readiness.

---

6. How Organizations Actually Use Logs

If telemetry volumes and costs are high, how is the data actually used?

Chronosphere's analysis of log data usage points to a simple pattern: the primary value of log data comes from **troubleshooting production systems and incident response**, not from continuous routine analysis. Troubleshooting and incident response were cited as top use cases by respondents, ahead of activities like performance tuning or business analytics.Chronosphere+1

In day-to-day operations, engineers and SREs often interact with aggregates and high-level metrics. They turn to logs when something is broken: to reconstruct a timeline, inspect stack traces, or validate hypotheses about what went wrong. In this sense, historical log archives act like an emergency black box in aviation: mostly ignored until disaster strikes, then suddenly invaluable.

This usage pattern explains both the tendency to "log everything just in case" and the feeling of waste. Organizations are paying to store enormous volumes of data that may never be touched—but when a "once in a quarter" or "once in a year" incident happens, they want to be certain that the necessary data is there.

The challenge therefore is not to eliminate logs, but to **change how value is extracted from them**, so that:

- Engineers do not spend hours manually combing through them;
- The organization can maintain necessary historical coverage without unsustainable cost.

---

7. Design Principles for a vNext Observability Layer

The situation described so far suggests that the next generation of observability should be built on different principles.

A vNext layer should:

1. **Prioritize Understanding Over Volume**
   The primary function is not to collect more data but to explain what is happening in terms that engineers and stakeholders can act on.

2. **Reuse Existing Investments**
   Most organizations already rely on platforms like Datadog, New Relic, CloudWatch, or Elastic. A new layer should **sit on top of these systems**, amplifying their value rather than replacing them.

3. **Reduce Cognitive Load in Incidents**
   The highest marginal value of observability occurs during incidents. A vNext system must be designed for those moments, turning chaotic telemetry into coherent narratives.

4. **Address Economic Reality**
   It should help organizations rationalize telemetry collection and storage, highlighting which data actually contributes to understanding and where there is waste.

5. **Close the Loop With Code**
   Observability should not stop at detecting and explaining failures. It should help **improve the code** and architecture that caused them.

WHAWIT has been conceived specifically around these principles.

---

8. WHAWIT as an Intelligent Observability Layer

WHAWIT's role in the stack is intentionally clear: it is **not** a replacement for Datadog, New Relic, or CloudWatch. Instead, it ingests logs and signals from these platforms and applies an intelligence layer that transforms telemetry into understanding.

At a high level, its architecture comprises:

- An **ingestion layer** that connects to existing log and metric sources;
- A **semantic processing layer** that normalizes and enriches logs;
- An **intelligence layer** that identifies anomalies, correlates signals, and infers likely causes;
- An **On-Call Hub** where incidents are presented as structured narratives rather than as raw dashboards;
- A **repository integration layer** that understands the codebase and connects runtime behavior to code evolution.

Because WHAWIT rides on top of existing observability tooling, it can be introduced incrementally. A startup already using a hosted log solution can adopt WHAWIT without ripping anything out. An enterprise with multiple overlapping tools can use WHAWIT as a unifying interpretive layer, helping to tame tool sprawl and extract more value from what they are already paying for.

---

9. The WHAWIT Intelligence Layer in Depth

The intelligence layer is where WHAWIT differentiates itself.

Instead of presenting a search bar and expecting humans to formulate queries under stress, WHAWIT continuously analyzes incoming logs and events, building an internal model of "normal" and "abnormal" patterns for each system. When an anomaly emerges—an error spike, a pattern of timeouts, a sequence of failures across services—it does not merely flag the condition; it works to **explain it**.

This explanation involves several kinds of reasoning:

- **Temporal reasoning:** What changed just before the issue started? Which services began exhibiting unusual behavior first?
- **Topological reasoning:** How are the affected services connected? Is there a downstream dependency failing upstream components?

- **Historical reasoning:** Have we seen a similar pattern before? If so, what was the root cause then?

The output is a synthesized incident summary in natural language, backed by pointers to the relevant log segments and metrics. Engineers can still drill down, but the initial cognitive burden of sifting through thousands or millions of lines of logs is removed.

From the perspective of MTTR, this is critical. Instead of spending the first 30–60 minutes of a major incident trying to understand what is happening, teams can often reach a workable hypothesis within minutes, reducing both the duration and chaos of the response.

---

10. The On-Call Hub: Where Understanding Meets Action

In practical terms, WHAWIT's intelligence layer surfaces through the **On-Call Hub**, a workspace designed for humans dealing with high-pressure situations.

The Hub provides:

- A **timeline** of key events as understood by the platform;
- A **summary** of the incident that can be read and understood quickly;
- Links into logs, metrics, and traces that support the summary;
- Context for escalations and handovers;
- A persistent record of the incident that can be revisited for post-mortems.

Because the Hub is built around explanation and narrative rather than raw charts, it supports teams whose members may not all be deep experts in every subsystem. New responders can be effective more quickly; senior engineers can focus on diagnosis and remediation rather than coordinating information.

---

11. The Autonomous Code Feedback Loop

Traditional observability ends when the incident is resolved. WHAWIT continues.

A central design choice in WHAWIT is its deep integration with **version control systems and CI/CD pipelines**. By analyzing repositories, commit histories, and structural patterns in the codebase, WHAWIT constructs an internal understanding of how the system is built and how it changes over time.

When incidents occur, the platform situates them within that context:

- It can identify which modules or services are statistically more failure-prone.
- It can see whether a recent commit touched the areas implicated by an incident.
- It can correlate regressions with specific change sets, not just symptoms.

From that vantage point, WHAWIT does not stop at saying *"this is probably where the problem is"*. It can propose **concrete improvements** to the code or configuration. Examples include more robust error handling, more defensive integration patterns, safer timeouts and retries, or test cases that cover uncovered failure scenarios. Where appropriate, WHAWIT can generate **candidate patches** and open pull requests automatically, positioning human engineers as reviewers rather than sole authors.

Over time, as patches are accepted or rejected and as the system evolves, WHAWIT learns which interventions are effective. The incident history becomes not just a set of post-mortems but a training corpus for the platform's internal models of resilience.

The result is a **closed loop**:

1. Production behavior generates telemetry.
2. WHAWIT interprets telemetry into understanding.
3. Understanding is mapped onto code.
4. Code is improved through suggestions and patches.
5. Future incidents are less likely or easier to resolve.

This concept—**autonomous reliability engineering**—turns observability from a read-only diagnostic surface into an active driver of software evolution.

---

12. ROI Modeling: Translating Capabilities into Economics

The question every buyer eventually asks is straightforward: *What does this do for our bottom line?*

While exact numbers will vary by organization, it is possible to outline the main levers where WHAWIT affects ROI:

1. **Reduction in Downtime (MTTR)**
   Consider a company with a handful of high-impact incidents per year, each costing several hundred thousand dollars per hour. If WHAWIT can help reduce MTTR for those incidents by even 30–50%, the annual savings may reach into the hundreds of thousands or millions, aligning with the patterns observed in studies where organizations with stronger observability practices report significantly less downtime and lower outage costs per hour.New Relic+1

2. **Recovery of Engineering Time**
   If a team of, say, 20 engineers currently spends 25% of its time on unplanned incident work and manual diagnostics, that represents 5 full-time equivalents of capacity. Even a modest 30–40% reduction in time spent on manual triage, enabled by WHAWIT's automated explanations and correlations, effectively returns 1.5–2 FTEs worth of productive time to

roadmap work.

3. **Improved Use of Existing Observability Investments**
   By placing an intelligence layer on top of platforms like Datadog or New Relic, WHAWIT increases their yield without requiring increased ingest. It can help organizations get more value from the data they are already paying to collect, and in some cases guide them toward more rational logging strategies that reduce redundant or low-value telemetry without sacrificing incident readiness. This aligns well with the desire—expressed by over 80% of surveyed DevOps teams—to control observability costs while maintaining or improving effectiveness.DevOps.com+1

4. **Compounding Improvements in Resilience**
   Through the autonomous code feedback loop, each resolved incident becomes an opportunity to measurably improve the system. Over a multi-year horizon, this compounding effect can reduce both the frequency and severity of outages, multiplying the value of the initial investment.

A formal ROI model can be parameterized for a given organization, but at a high level, WHAWIT's value comes from this combination: **fewer and shorter outages, less engineering toil, better use of existing tools, and continuous reinforcement of code-level resilience**.

---

13. Startups vs Enterprise: Different Scales, Similar Pain

The challenges described in this paper are not limited to large enterprises.

- **Startups and scale-ups** often operate under tight budgets and lean teams. They cannot afford to hire a large SRE group, but they still face customer-visible outages and the expectation of 24×7 availability. For them, the ability to compress MTTR and reduce the operational tax on a small engineering team can be existential. The telemetry explosion hits them early: even a moderately complex SaaS can generate hundreds of gigabytes of logs per month, and mistakes in log configuration can produce surprise bills from observability vendors.Chronosphere+1

- **Enterprises** have the scale to invest in sophisticated observability stacks, but they suffer from complexity and cost. They often run multiple overlapping tools, large historical log clusters, and strict retention policies for compliance. Leadership is increasingly concerned with the cost of these systems; surveys show that a vast majority of large organizations believe they are overspending on observability and anticipate scrutiny to cut costs.DevOps.com+1 At the same time, the cost of downtime for a large enterprise is so high that cutting corners on visibility is not an option.

In both contexts, WHAWIT addresses the same underlying pain: **current observability is too manual, too expensive, and too detached from code**. The platform is designed to be valuable whether a company has 5 engineers or 500.

14. Conclusion: From Monitoring to Understanding to Improvement

Over the past decade, monitoring and observability have progressed from simple uptime checks to sophisticated, multi-signal telemetry platforms. But the core workflow of incident response—the human labor of making sense of what the data means—has changed much less.

The research summarized here points to three converging realities:

- Downtime is extraordinarily expensive, regardless of company size.
- Telemetry volumes and observability costs are rising faster than budgets.
- Most organizations still depend on manual effort to interpret incidents and translate them into lasting improvements.

WHAWIT is designed as a response to that reality. By sitting on top of existing observability stacks, interpreting telemetry semantically, supporting on-call engineers with real explanations rather than raw charts, and closing the loop into code, it aims to shift observability from **data exposure to understanding and autonomous improvement**.

In doing so, WHAWIT reframes observability as a strategic capability: not just a way to see what is happening, but a way to continuously learn from and harden the systems that matter most.