Enterprise Strategy Group™

BRIEF

APRIL 2025

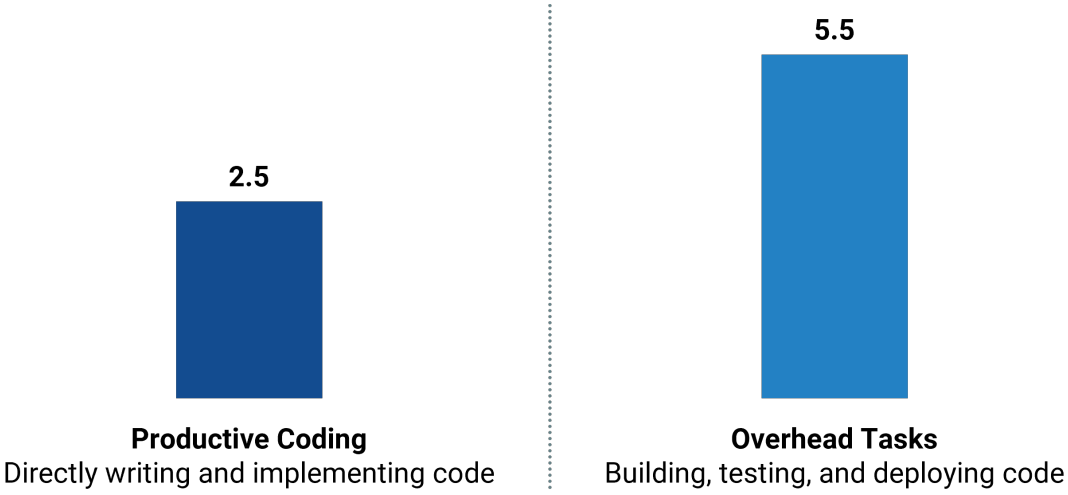
Observability in Action: Enhancing Developer Productivity in Real-world Scenarios

Torsten Volk, Principal Analyst

Overview: Developer Challenges

Based on research from Enterprise Strategy Group, now part of Omdia, application developers spend only 35% of their time, a little more than two and a half out of eight work hours every day, on productive coding.[1] This means that the remaining five and a half hours go to overhead tasks such building, testing, deploying, and troubleshooting code for different on-premises and cloud infrastructure; defining, provisioning, monitoring, and optimizing cloud-specific infrastructure- and platform resources; using AI to personalize experiences; and making everything work in heterogeneous application environments.

Figure 1. Time Allocation of Application Developers



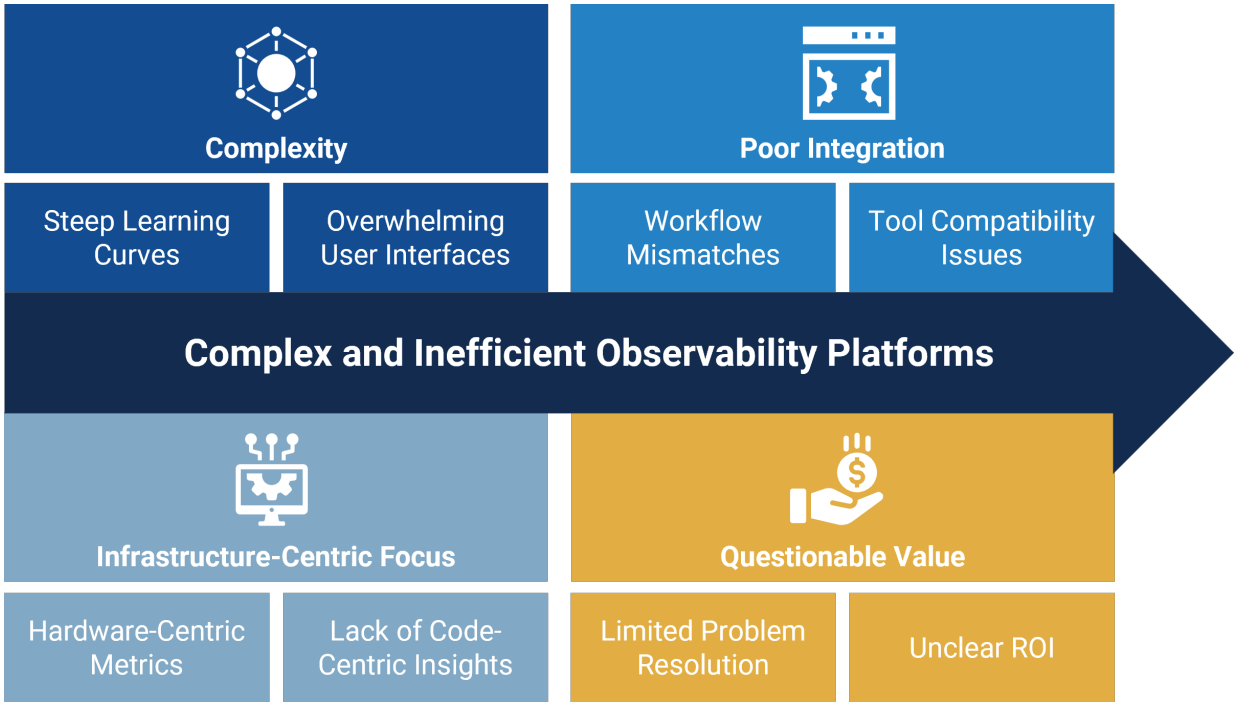
Source: Enterprise Strategy Group, now part of Omdia

Reclaiming these less-productive hours has become a critical factor for business success, as productive coding enables organizations to deliver more customer value faster, and at higher quality, without hiring more developers. At the same time, freeing developers from tedious overhead tasks increases job satisfaction and motivation to deliver the best possible products to their clients.

To address these challenges, innovative solutions like Dynatrace Live Debugging are emerging as key enablers of developer productivity. By providing real-time insights into application performance without disrupting the live system, such tools can significantly reduce the time spent on troubleshooting and debugging, enabling developers to focus more on coding and delivering value.

[1] Source: Enterprise Strategy Group Research Report, *Application Modernization and the Role of Platform Engineering*, October 2024.

Figure 2. Observability Challenges for Developers



Source: Enterprise Strategy Group, now part of Omdia

Observability can directly enhance developer productivity by providing a clear understanding of how applications are used in real life, how these usage patterns affect infrastructure and business, and how code changes impact application health and performance in complex hybrid environments. Ideally, developers can observe in real time how each line of code interacts with the rest of the application stack to proactively address any performance, reliability, and user experience issues.

However, organizations often struggle to convince developers to use observability platforms because they are perceived as complex and infrastructure-centric and are poorly integrated with current workflows. Dynatrace Live Debugging addresses these challenges by offering an intuitive and code-centric approach to observability, enabling developers to debug live systems without disrupting them.

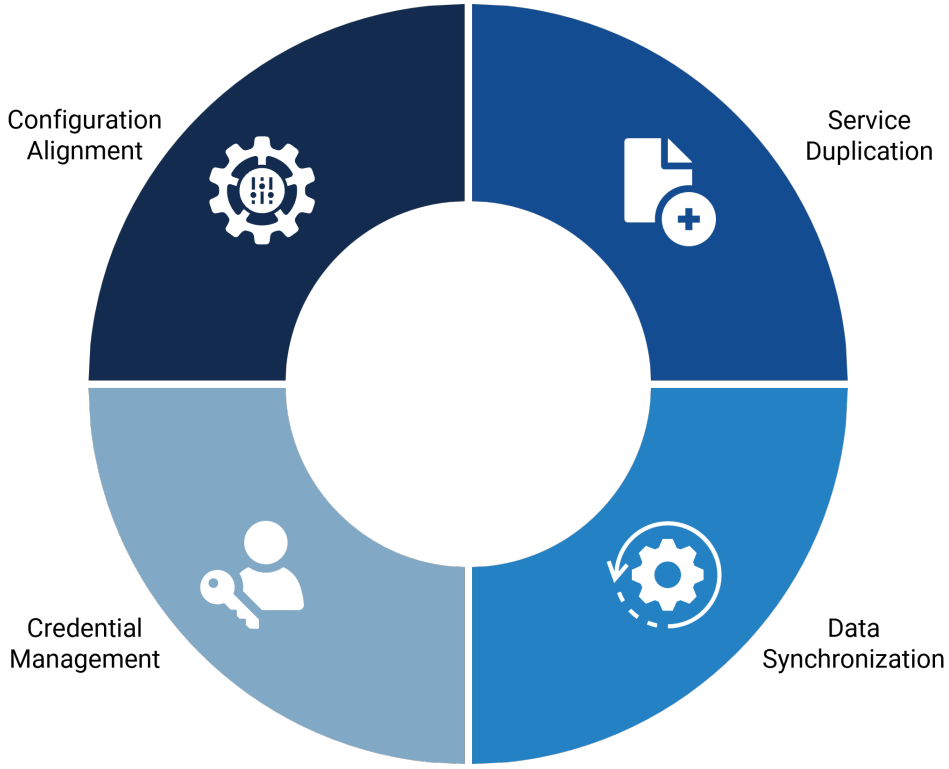
Debugging Complex Production Applications Is Hard

The value of an observability platform for software developers is best explained by looking at an example of how developers work and at the daily problems they encounter: Assume a new code push to production has resulted in the deterioration of response times of an application's API. Debugging the live system is typically not possible due to the disruptive character of the debugging process. Even if disruptions were acceptable, the application might just reroute user requests elsewhere, withholding critical user interactions data.

Due to the inability to debug production applications, developers need to replicate their application stacks within a staging environment. This is difficult in modern distributed microservices applications, as these often depend on numerous external components that cannot easily be replicated within a staging environment. For example, an e-commerce platform might rely on third-party payment gateways, inventory systems, and shipping providers. Reproducing those exact dependencies and usage patterns in a staging environment would require not only duplicating each external service but also synchronizing real-time data, credentials, and configurations—an often prohibitively complex and expensive task.

Additionally, simulating user interactions in a realistic manner is tricky. It is not just about marching through the "golden path" that developers meticulously map out. Certain issues might only surface in edge cases when mobile users struggle with flaky connectivity or when external microservices use malformed request headers for their API calls. Also, shared microservices that seem fine under normal conditions might buckle under sudden usage spikes.

Figure 3. Challenges of Staging Environments



Source: Enterprise Strategy Group, now part of Omdia

Dynatrace Live Debugging: Non-breaking Breakpoints for Production Environments

Dynatrace Live Debugging enables developers to not worry about any of the challenges related to cloning production environments and replicating end-user behavior because they are now able to look directly into the live system without interrupting the application.

For example, to diagnose seemingly random latency spikes in the checkout process of a shopping cart, developers can set so-called non-breaking breakpoints at the beginning and at the end of all relevant processes across different microservices. They will then receive a detailed breakdown of how long each step of the overall checkout process took, for all of the different user transactions, in real time. In addition to this breakdown, the live debugger collects potentially relevant transaction parameters, such as the number of items in the cart, the value of those items, the time of day, the geographic location, etc. While developers might spot clear correlations (e.g., the more items in the cart, the worse the latency of that specific transaction), some AI magic is needed for actual root cause analysis. This is simply due to the large number of additional parameters collected for each transaction and the even larger number of permutations of these parameters that could constitute the root cause of the latency issue.

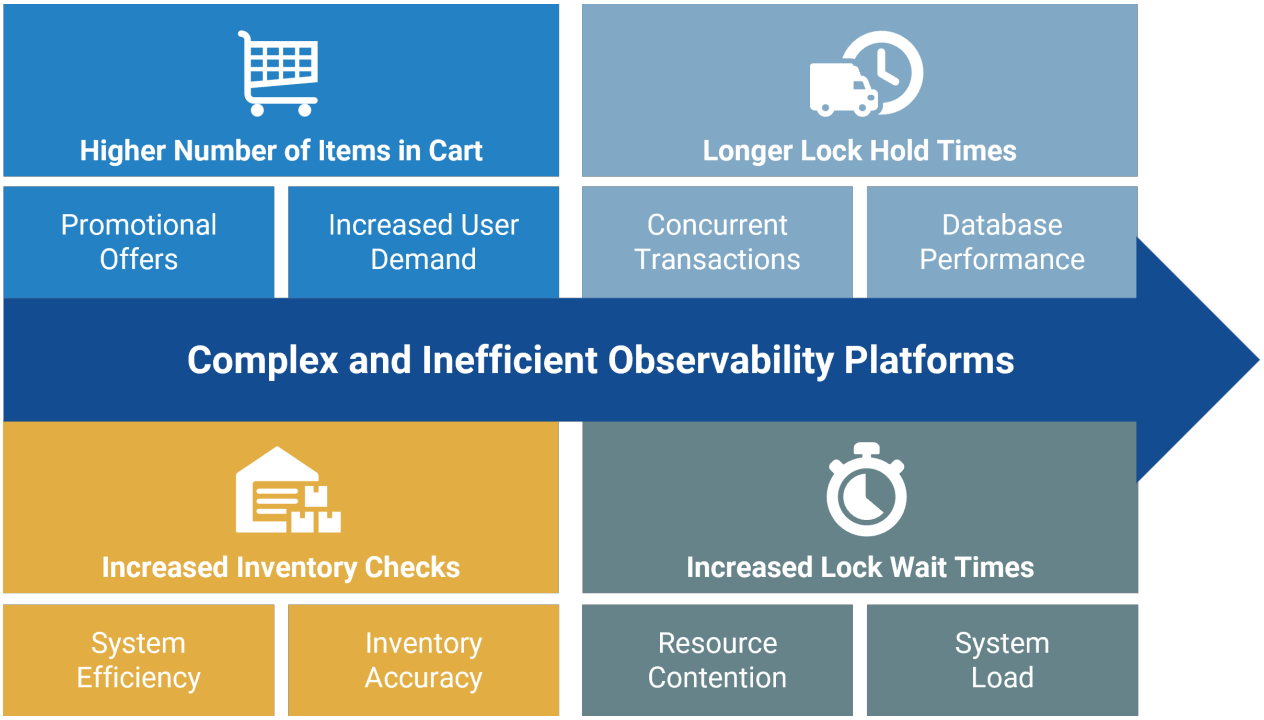
Figure 4. Live Debugging Enhances the Checkout Process

Source: Enterprise Strategy Group, now part of Omdia

The following wide range of factors are possible causes of the latency spikes:

- **Database connection pool metrics:**
  - Active connections: Number of active database connections at any given time.
  - Idle connections: Number of idle connections waiting to be reused.
  - Connection wait time: Time spent waiting for a connection to become available.
- **Network request details:**
  - Request headers: Errors related to Authorization, Content-Type, etc.
  - Response status codes: Transmission errors.
  - Response body: Size of content returned by external services.
- **Memory and resource usage:**
  - Heap memory usage: Amount of allocated versus used memory.
  - Garbage collection cycles: Frequency of GC cycles are causing.
  - Thread pool sizes: Sizing of thread pools for concurrent tasks.
- **Cache hit rates:**
  - Cache hits: Number of times cached data is successfully retrieved.
  - Cache misses: Number of times data must be fetched from a slower source.
- **Lock contention:**
  - Lock wait time: Time spent waiting for locks to be released.
  - Lock hold time: Duration locks are held, potentially blocking other threads.
- **Transaction logs:**
  - Transactions: Number and requirements of transactions in progress.
  - Transaction Status: Number of transactions that are committed, rolled back, or pending.
- **Queue Depths:**
  - Message queue length: Number of messages waiting to be processed.
  - Queue processing time: Time taken to process messages in the queue.

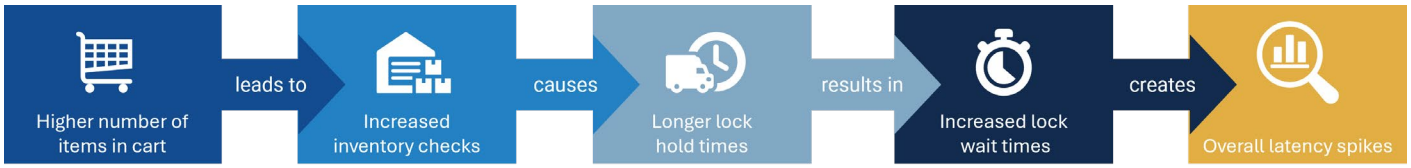
Figure 5. Analyzing Causes of Latency Spikes in E-commerce



Source: Enterprise Strategy Group, now part of Omdia

By integrating all telemetry data into Dynatrace, powered by its Davis AI engine, which combines causal AI, predictive AI, and generative AI, Dynatrace can automatically identify the root cause of issues without requiring user input or a background in data science. For instance, an unexpected surge in active database connections might result in longer connection wait times, ultimately causing latency in inventory checks. Davis AI can automatically pinpoint the root cause of such latency spikes by analyzing temporal dependencies and contextual information within the time-series data. Based on these insights, the relationships could look like this:

Figure 6. Sample Root Cause Analysis



Source: Enterprise Strategy Group, now part of Omdia

Now, causal AI and generative AI can collaborate to make recommendations for resolving the issue. Ultimately, the observability platform could automatically implement predictive or incident remediation steps with Dynatrace AutomationEngine, including verification of resolutions. Developers may then choose to continue the data collection, at least for edge cases, to receive early warnings in case the problem returns.

Live Debugging Rocks: Observability for Developers Is Key

The combination of Live Debugging with causal, predictive, and generative AI, combined with the comprehensive mapping of all relationships between the infrastructure, application, user experience and business layer of applications across the enterprise into one data lake (Grail) creates a strong platform for unlocking a significant share of the 65% of developer productivity that is currently spent on non-value-generating tasks. The Dynatrace Observability for Developers platform focuses on addressing the key reasons why developers often do not take advantage of observability. It provides a set of developer-centric capabilities that are easy to use, integrate well within developer tools, and show a clear focus on optimizing application code.

About Enterprise Strategy Group
Enterprise Strategy Group, now part of Omdia, provides focused and actionable market intelligence, demand-side research, analyst advisory services, GTM strategy guidance, solution validations, and custom content supporting enterprise technology buying and selling.

contact@esg-global.com
www.esg-global.com