On the Relationship Between AIOps and Systems Engineering

Wahab Hamou-Lhadj, PhD
Concordia University
Montréal, QC, Canada
wahab.hamou-lhadj@Concordia.ca

NASA JPL, Pasadena, CA, USA
April 10, 2023
What is AIOps?

- AIOps is the application of **AI to enhance IT** operations
  - An important enabler of digital transformation

- AIOps relies heavily on **observability** mechanisms to collect operational data
  - Data is collected automatically from devices, IT platforms, applications with no direct user intervention

- Three main **applications**:
  - Improving quality of service
  - Regulatory compliance
  - Operational intelligence
Scope of AIOps

Industries

- Telecom
- Healthcare
- Energy
- Defense
- Manufacturing
- Finance
- Aerospace
- Education
- and more

Technology and Processes

- Fault Diagnosis & Repair
- Failure Prediction
- Operational Intelligence
- Anomaly Detection
- Cybersecurity
- Compliance Management
- Data Merging, Fusion, Abstraction
- Storage, Processing, Data Analytics, AI, Visualization, Etc.
- Logs, Traces, Metrics, Incident Reports
- System of Systems
- E.g. Applications
Why AIOps?

- Operational complexity of today’s highly distributed and dynamic systems
  - A 2022 study by AppDynamics shows that 91% of participants believe that gaining full observability into their systems would be revolutionary for their business¹
  - A VMware report shows that traditional monitoring tools are not enough to understand today’s complexity of large-scale systems

- Panoply of tools
  - A typical company uses hundreds of tools for all sort of IT-related tasks

- Challenges hiring and retaining workforce
  - 4.3 million people quit jobs in August 2021 — about 2.9 percent of the workforce. The NY Times, 2021³
Software Observability

- **In control theory:**
  - **Observability** is “a measure of how well internal states of a system can be inferred from knowledge of its external outputs” [Wikipedia]

- **Software Observability:**
  - A set of end-to-end techniques and processes that allow us **to reason** about **what a software system is doing and why** by analyzing its external outputs.
Monitoring vs Observability

- Monitoring:
  - Tracks known metrics and raises alerts when thresholds are not met (e.g., 4 golden signals of Google SRE: latency, traffic, errors, and saturation)
  - Answers the question: “how is the system doing?”
  - Helps diagnose known problems

- Observability:
  - Answers the question: “what is the system doing and why?”
  - Enables to reason about the system by observing its outputs
  - Helps diagnose known and unknown problems
Building Blocks

Distributed and Complex Systems in Operation

Data Collection → Execution Profile → Analytics

Offline and/or real-time analytics
Telemetry Data

- **Logs:**
  - Records of events generated from logging statements inserted in the code to track system execution, errors, failures, etc.
  - Different types of logs: system logs, application logs, event logs, etc.

- **Traces:**
  - Records of events showing execution flow of a service or a (distributed) system with causal relationship
  - Require additional instrumentation mechanisms

- **Profiling Metrics:**
  - Aggregate measurements over a period of time (e.g., CPU usage, number of user requests, etc.)
Challenges

▪ Standards and Best Practices:
  ▪ Lack of guidelines and best practices for logging, tracing, and profiling
  ▪ Lack of standards for representing logs, traces, and metrics (not the OpenTelemetry initiative)

▪ Data Characteristics
  ▪ Mainly unstructured data
  ▪ Size is a problem
  ▪ Not all data is useful
  ▪ High velocity
Challenges

- **Analytics and Tools:**
  - Mainly descriptive analytics
  - Predictive analytics not fully explored
  - Mainly offline analysis techniques
  - Lack of usable end-to-end observability tools

- **Cost and Management Aspects**
  - Cost vs. benefits not well understood
  - No clear alignment of observability with other initiatives
  - Roles and responsibilities are not well defined
Current Projects

- **ULP: Universe Log Parser**
  - A unified framework to extract structured information from unstructured logs using ML

- **Incident Report Triaging**
  - A set of techniques for reducing lead time of fixing crashed and system failures

- **TotalADS: Anomaly Detection**
  - An adaptable anomaly detection framework based on Boolean combination of classifiers

- **ClusterCommit: Predicting buggy code commits using AI**
  - A framework for predicting bugs as developers commit code based on historical commits
ULP: Universe Log Parser

Raw Logs → ULP → Log Templates

Log Templates
Incident Reports Handling Process

Incident Reports

1\textsuperscript{st} line support

2\textsuperscript{nd} line support

3\textsuperscript{rd} line support

Development Teams

Log data
Memory dumps
AI-Driven Incident Report Triaging

- Building datasets of incident reports
- Detection of Duplicate Reports
- Prediction of Report Severity
- Automatic Routing of Incident Reports
TotalADS: Total Anomaly Detection
System Architecture

Software-Intensive Infrastructures
- Machine Data Management
  - Controller
  - Loading
  - Streaming

Boolean Combination Engine
- SVM
- NN
- KSM
- RNN
- Others

Control Center
- Plots
- Controllers
- Statistics
- Reports
- Analysis

Data Centers
Radio Stations
Smart Grids
IoT Devices

IBM CASCON
PEOPLE’S CHOICE
AWARD
What is the place of system modeling in AIOps?

- Emerging technologies require system-wide observability
  - Industry 4.0, CPS, autonomous vehicles, IoT

- A model of the system in operation (digital twin?)
  - A model of a system in operation can guide analysis for current and future versions of the system

- Experimental vs. formal analysis
  - System engineering offers the level of rigor needed for analysis that is not found in experimental development
Brining observability to early stages of the SDLC using Sys Eng.

- Bringing observability **to early stages** of the development lifecycle

- **Cost of observability** can be assessed during project planning
Brining observability to early stages of the SDLC using Sys Eng.

- Observability as a non-functional requirement
- What aspects of system functional requirements should be observable and how?

Diagram:
- Requirement Analysis
  - Maintenance
  - Design
  - Observability
  - Deployment
  - Implementation
  - Testing
Brining observability to early stages of the SDLC using Sys Eng.

- Support of observability at the architectural level
- Detailed design for observability
- Observability patterns and best practices

![Diagram showing the lifecycle stages: Requirement Analysis, Design, Implementation, Testing, Deployment, Maintenance, and Observability, with arrows indicating the flow.](image-url)
Brining observability to early stages of the SDLC using Sys Eng.

- What, where, and how to log and/or trace?
- Use of libraries and frameworks
- Patterns and best practices
Brining observability to early stages of the SDLC using Sys Eng.

- Testing and inspection strategies for logging/tracing code

Diagram:
- Requirement Analysis
- Design
- Maintenance
- Deployment
- Implementation
- Observability
- Testing
Brining observability to early stages of the SDLC using Sys Eng.

- Deployment, configuration, and maintenance aspects of observability code such as updates, performance analysis, testing, persistence, etc.
Open Questions?

- What should a model of a system in operation look like?
- Which aspects of MBSE we can easily leverage to support system-wide observability and AIOps?
- Should we start talking about model-driven AIOps?
- Is ontology modeling and analysis the way to go?
Contact Information

Wahab Hamou-Lhadj, PhD, ing.
Concordia University
wahab.hamou-lhadj@concordia.ca
http://www.ece.concordia.ca/~abdelw
References

1. Linda Tong, "Momentum is building on the journey to observability," online: https://www.appdynamics.com/blog/full-stack-observability/momentum-is-building-on-the-journey-to-observability/

2. Stela Udovicic, "The State of Observability 2021: Key Findings," Retrieved online: https://tanzu.vmware.com/content/blog/the-state-of-observability-2021-key-findings