On the Relationship Between AIOps and Systems Engineering

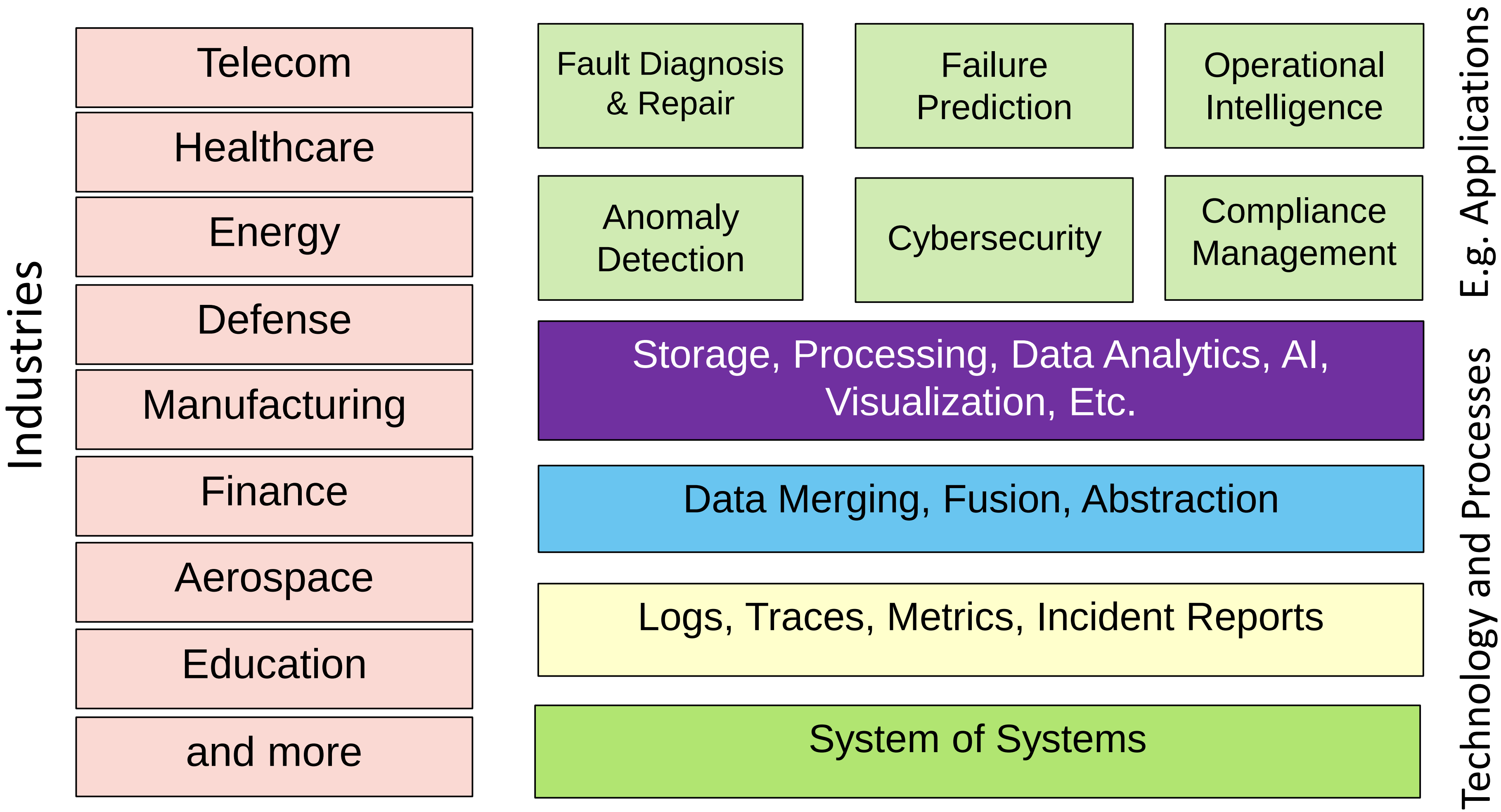
Wahab Hamou-Lhadj, PhD
Concordia University
Montréal, QC, Canada
wahab.hamou-lhadj@Concordia.ca

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



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 business[1]
  - 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, 2021[3]

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]

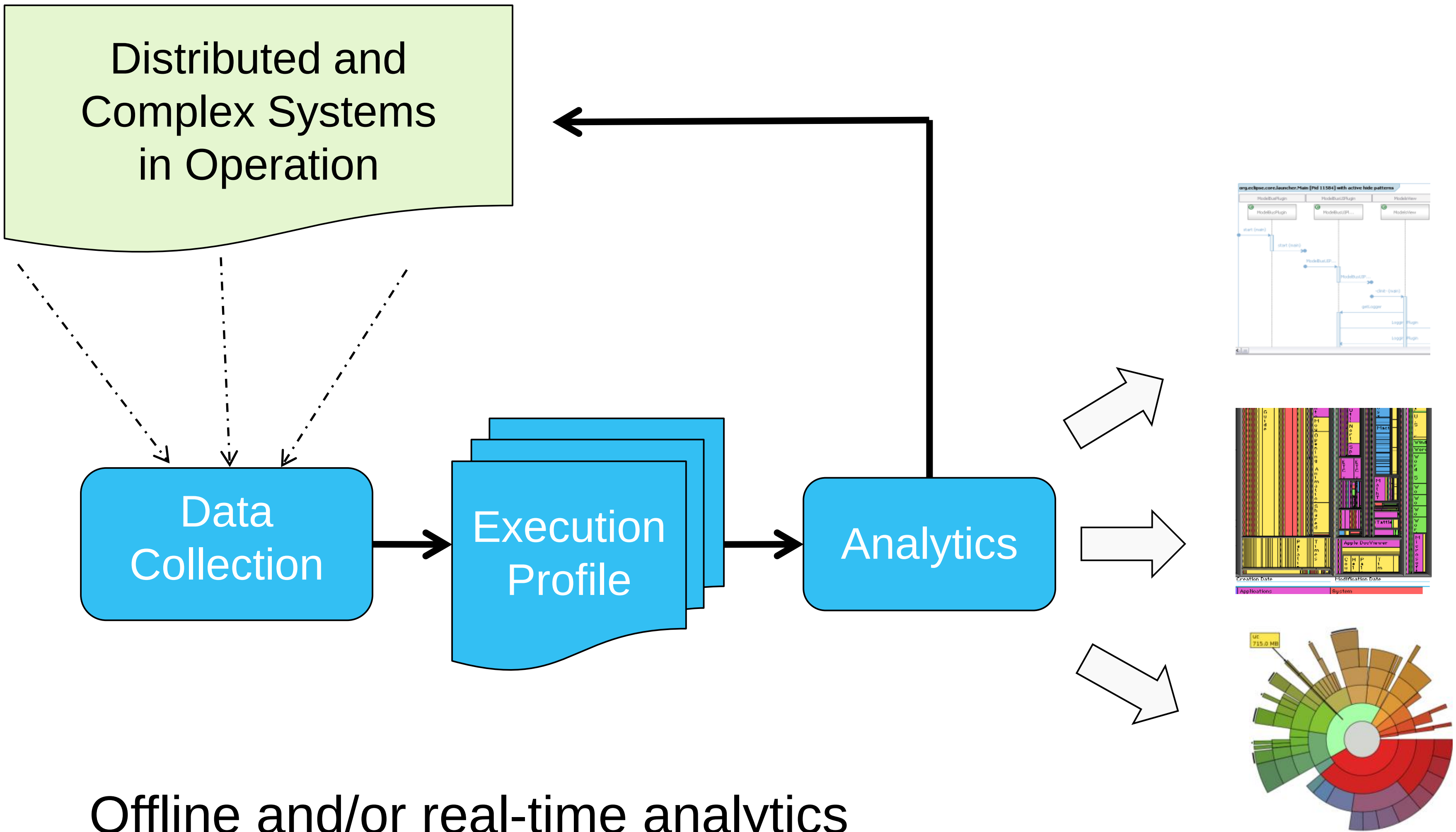
- **Software Observability:**
  - A set of end-to-end techniques and processes that allow us **to reason** about **what a software system is doing and why** by analyzing its external outputs.

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



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

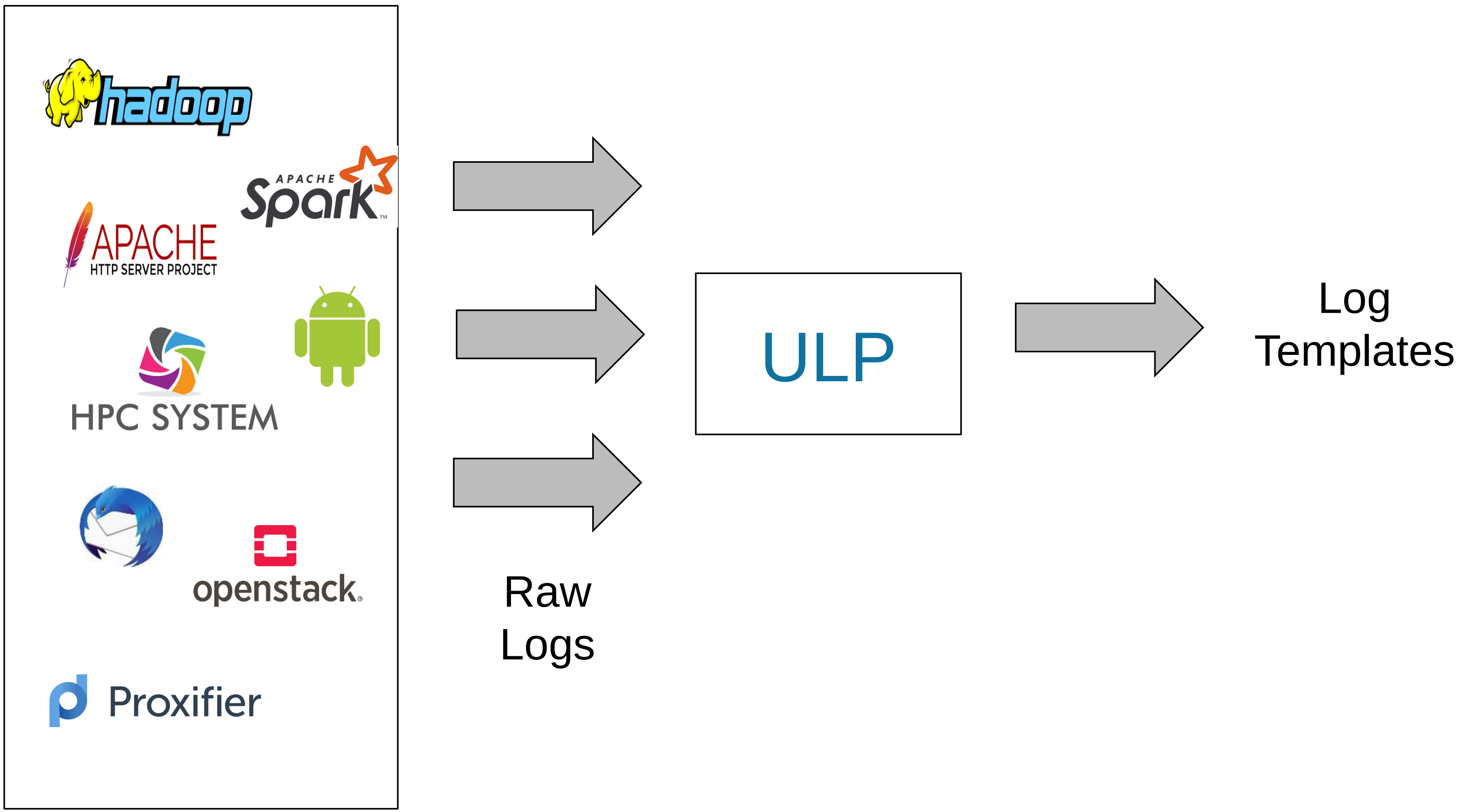
- **Analytics and Tools:**
  - Mainly descriptive analytics
  - Predictive analytics not fully explored
  - Mainly offline analysis techniques
  - Lack of usable end-to-end observability tools

- **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

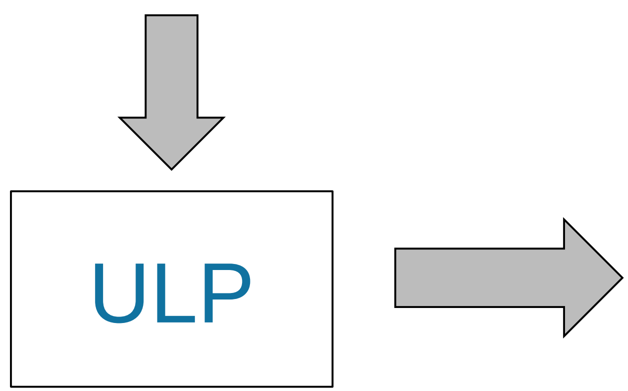
Current Projects

- **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

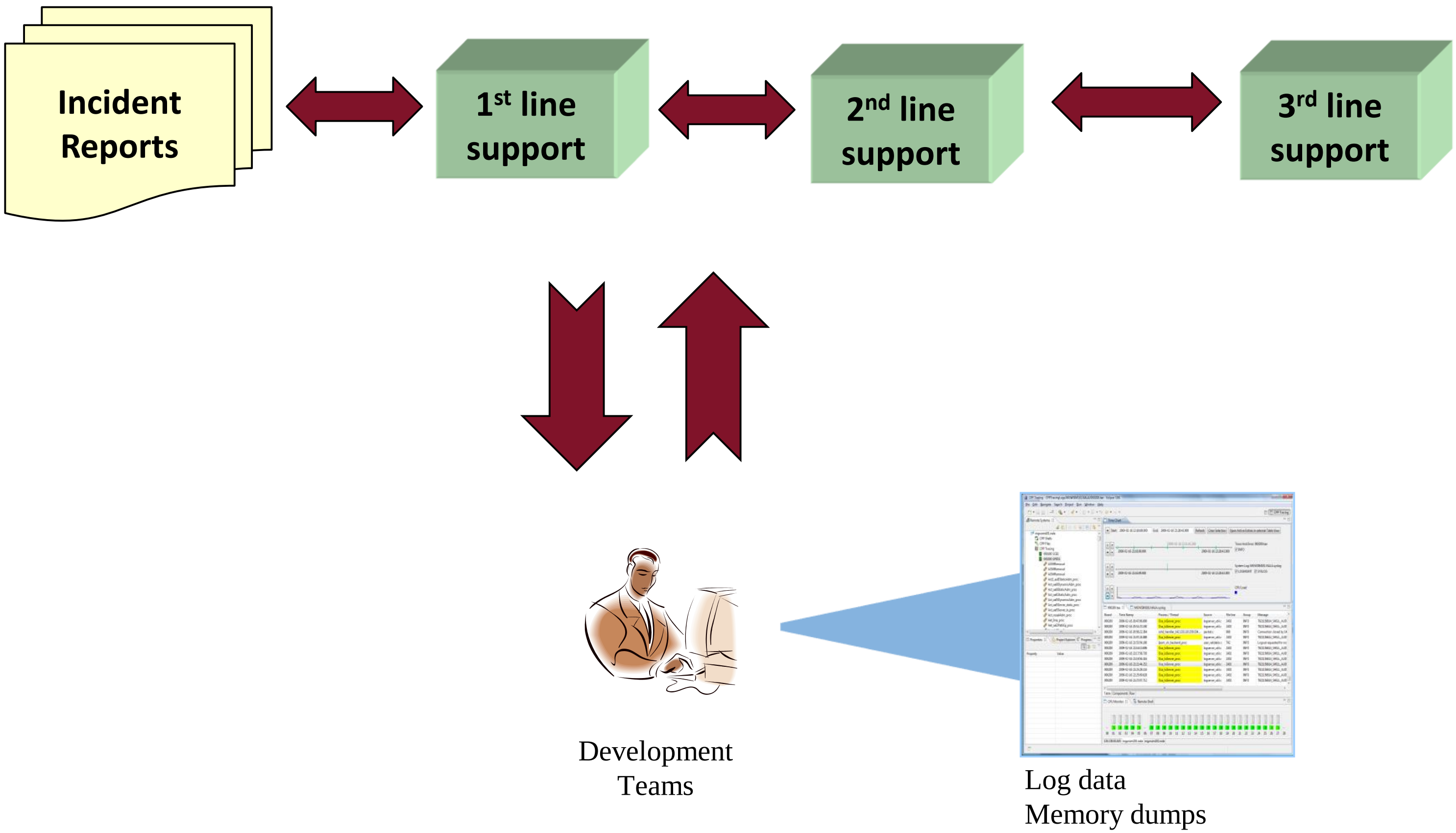


```
081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_38865049064139660 terminating
081109 203807 222 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_-6952295868487656571 terminating
081109 204005 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.73.220:50010 is added to blk_7128370237687728475
081109 204015 308 INFO dfs.DataNode$PacketResponder: PacketResponder 2 for block blk_8229193803249955061 terminating
081109 204106 329 INFO dfs.DataNode$PacketResponder: PacketResponder 2 for block blk_-6670958622368987959 terminating
081109 204132 26 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.43.115:50010 is added to blk_3050920587428079149
081109 204324 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.203.80:50010 is added to blk_7888946331804732825
081109 204453 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.11.85:50010 is added to blk_2377150260128098806
081109 204525 512 INFO dfs.DataNode$PacketResponder: PacketResponder 2 for block blk_572492839287299681 terminating
081109 204655 556 INFO dfs.DataNode$PacketResponder: Received block blk_3587508140051953248 of size 67108864 from /10.251.42.84
081109 204722 567 INFO dfs.DataNode$PacketResponder: Received block blk_5402003568334525940 of size 67108864 from /10.251.214.112
081109 204815 653 INFO dfs.DataNode$DataXceiver: Receiving block blk_5792489080791696128 src: /10.251.30.6:33145 dest: /10.251.30.6:50010
081109 204842 663 INFO dfs.DataNode$DataXceiver: Receiving block blk_1724757848743533110 src: /10.251.111.130:49851 dest: /10.251.111.130:50010
081109 204908 31 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.110.8:50010 is added to blk_8015913224713045110
081109 204925 673 INFO dfs.DataNode$DataXceiver: Receiving block blk_-5623176793330377570 src: /10.251.75.228:53725 dest: /10.251.75.228:50010
081109 205035 28 INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: /user/root/rand/_temporary/_task_200811092030_0001_m_000590_0/part-00590.
081109 205056 710 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_5017373558217225674 terminating
081109 205157 752 INFO dfs.DataNode$PacketResponder: Received block blk_9212264480425680329 of size 67108864 from /10.251.123.1
081109 205315 29 INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: /user/root/rand/_temporary/_task_200811092030_0001_m_000742_0/part-00742.
081109 205409 28 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.111.130:50010 is added to blk_456843418269316554
081109 205412 832 INFO dfs.DataNode$PacketResponder: Received block blk_-5704899712662113150 of size 67108864 from /10.251.91.229
081109 205632 28 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.74.79:50010 is added to blk_-4794867979917102672
081109 205739 29 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.38.197:50010 is added to blk_8763662564934652249
081109 205742 1001 INFO dfs.DataNode$PacketResponder: Received block blk_-5861636720645142679 of size 67108864 from /10.251.70.211
081109 205746 29 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.74.134:50010 is added to blk_7453815855294711849
081109 205749 997 INFO dfs.DataNode$DataXceiver: Receiving block blk_-28342503914935090 src: /10.251.123.132:57542 dest: /10.251.123.132:50010
081109 205754 952 INFO dfs.DataNode$PacketResponder: Received block blk_8291449241650212794 of size 67108864 from /10.251.89.155
081109 205858 31 INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: /user/root/rand/_temporary/_task_200811092030_0001_m_000487_0/part-00487.
081109 205931 13 INFO dfs.DataBlockScanner: Verification succeeded for blk_-4980916519894289629
081109 210022 1110 INFO dfs.DataNode$PacketResponder: Received block blk_-5974833545991408899 of size 67108864 from /10.251.31.180
081109 210037 1084 INFO dfs.DataNode$DataXceiver: Receiving block blk_-5009020203888190378 src: /10.251.199.19:52622 dest: /10.251.199.19:50010
081109 210248 1138 INFO dfs.DataNode$PacketResponder: Received block blk_6921674711959888070 of size 67108864 from /10.251.65.203
081109 210407 33 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.7.244:50010 is added to blk_5165786360127153975
081109 210458 1278 INFO dfs.DataNode$DataXceiver: Receiving block blk_2937758977269298350 src: /10.251.194.129:37476 dest: /10.251.194.129:50010
081109 210551 32 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.6.191:50010 is added to blk_6738257740739667l0 s:
081109 210637 1283 INFO dfs.DataNode$PacketResponder: Received block blk_-7526945448667194862 of size 67108864 from /10.251.203.80
081109 210656 1334 INFO dfs.DataNode$PacketResponder: Received block blk_-2094397855762091248 of size 67108864 from /10.251.126.83
```

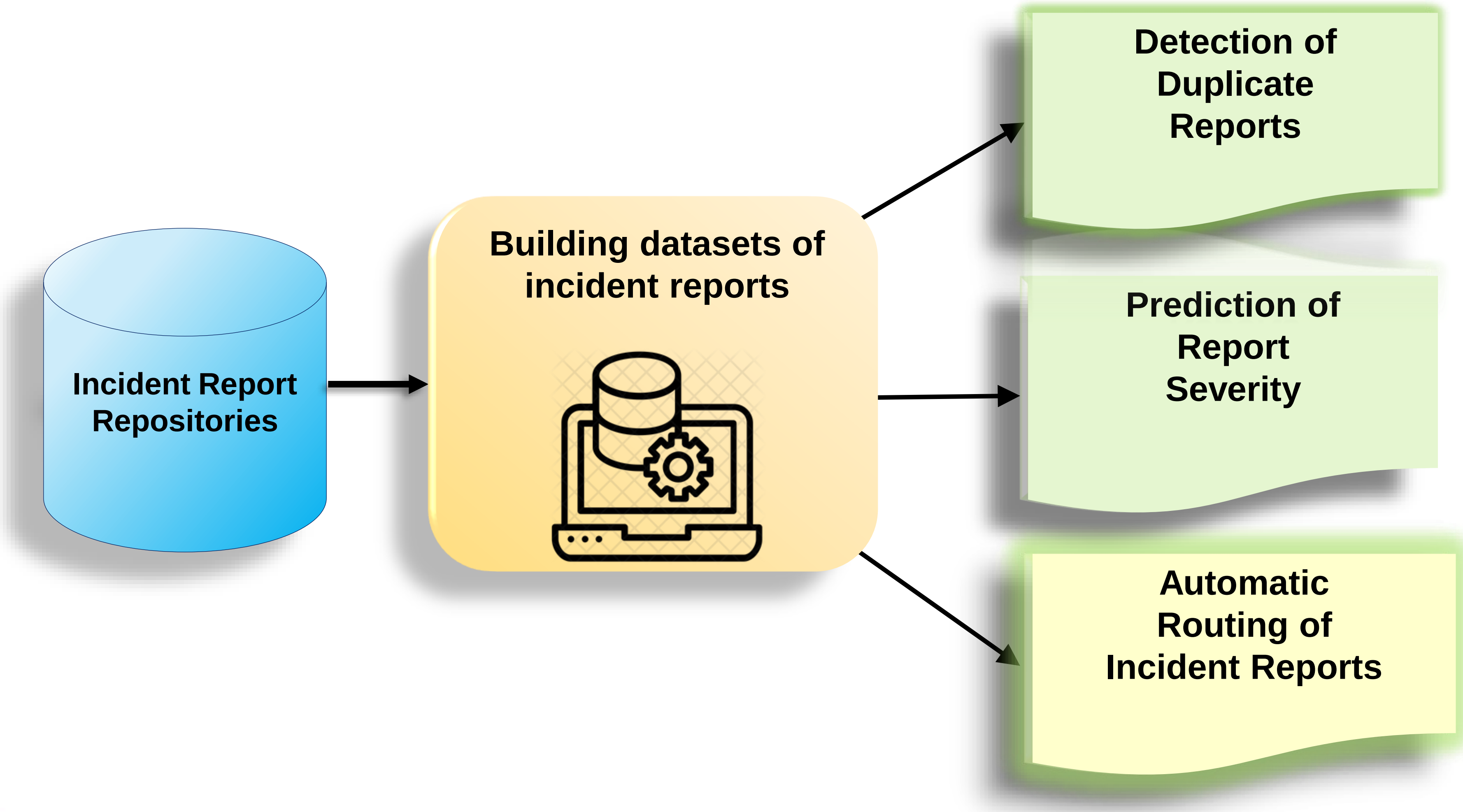


Log Template	Count
[block namesystem.addstoredblock: blockmap updated: <*> is added to <*> size <*>]	314
[packetresponder <*> for block <*> terminating]	311
[received block <*> of size <*> from /<*>]	292
[receiving block <*> src: /<*> dest: /<*>]	292
[deleting block <*> file /<*>]	263
[block namesystem.delete: <*> is added to invalidset of <*>]	224
[block namesystem.allocateblock: /<*>. <*>]	115
[<*> served block <*> to /<*>]	80
[<*>got exception while serving <*> to /<*>:]	80
[verification succeeded for <*>]	20
[received block <*> src: /<*> dest: /<*> of size <*>]	2
[block ask <*> to delete <*> <*> <*> <*> <*> <*> <*> <*> <*> <*> <*> <*> <*> <*> <*>	2
[block ask <*> to delete <*>]	2
Grand Total	**2000**

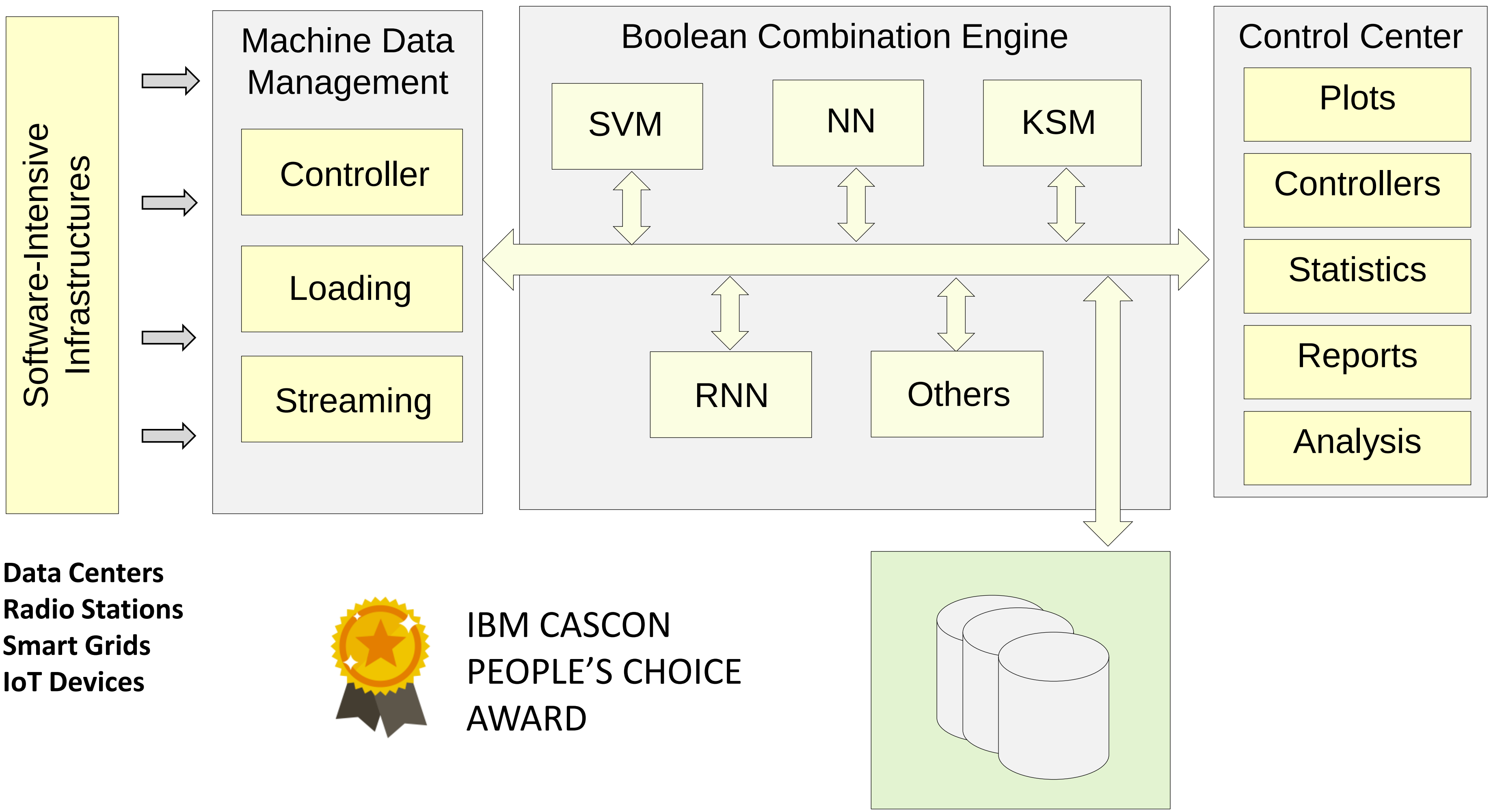
Incident Reports Handling Process



AI-Driven Incident Report Triaging



TotalADS: Total Anomaly Detection System Architecture



Data Centers
Radio Stations
Smart Grids
IoT Devices

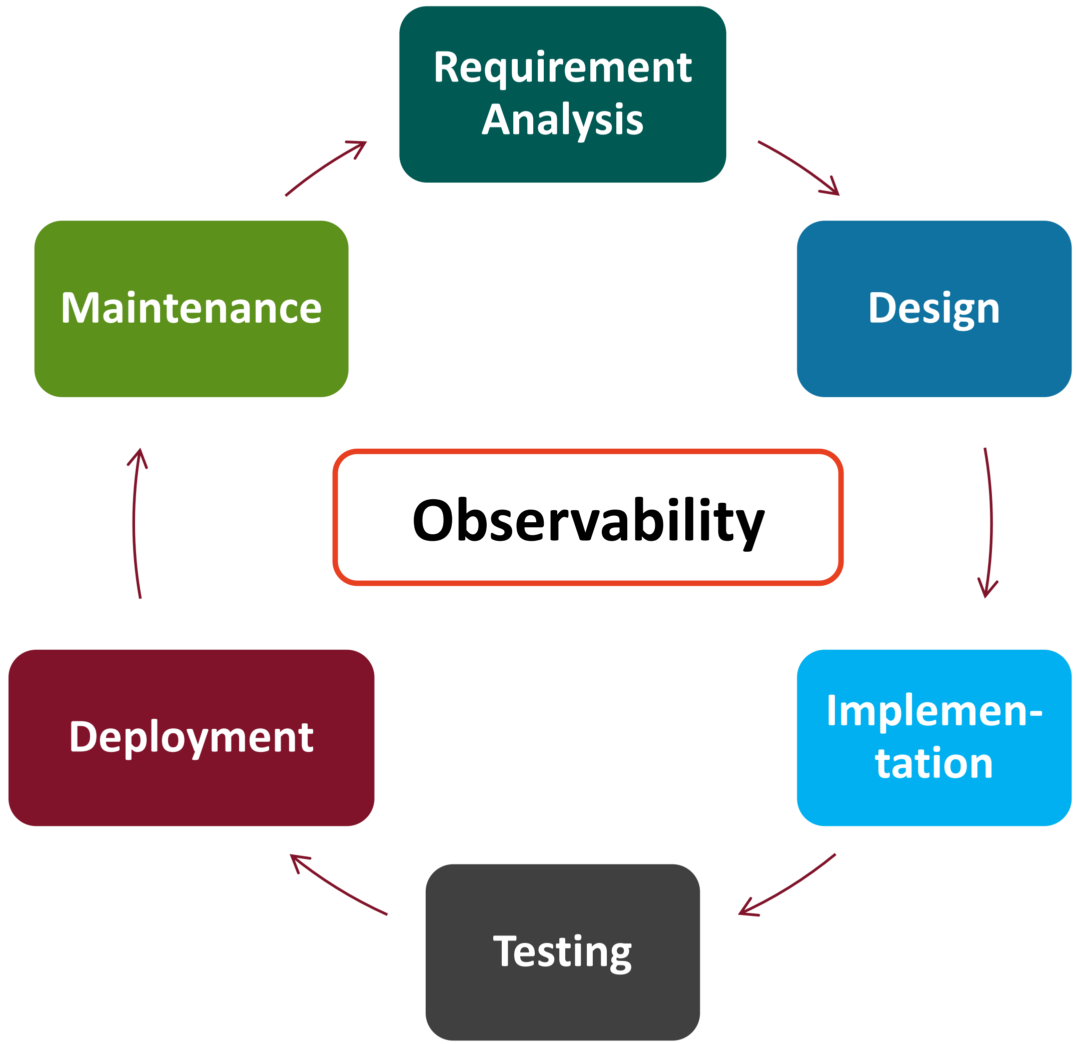
IBM CASCON PEOPLE'S CHOICE AWARD

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

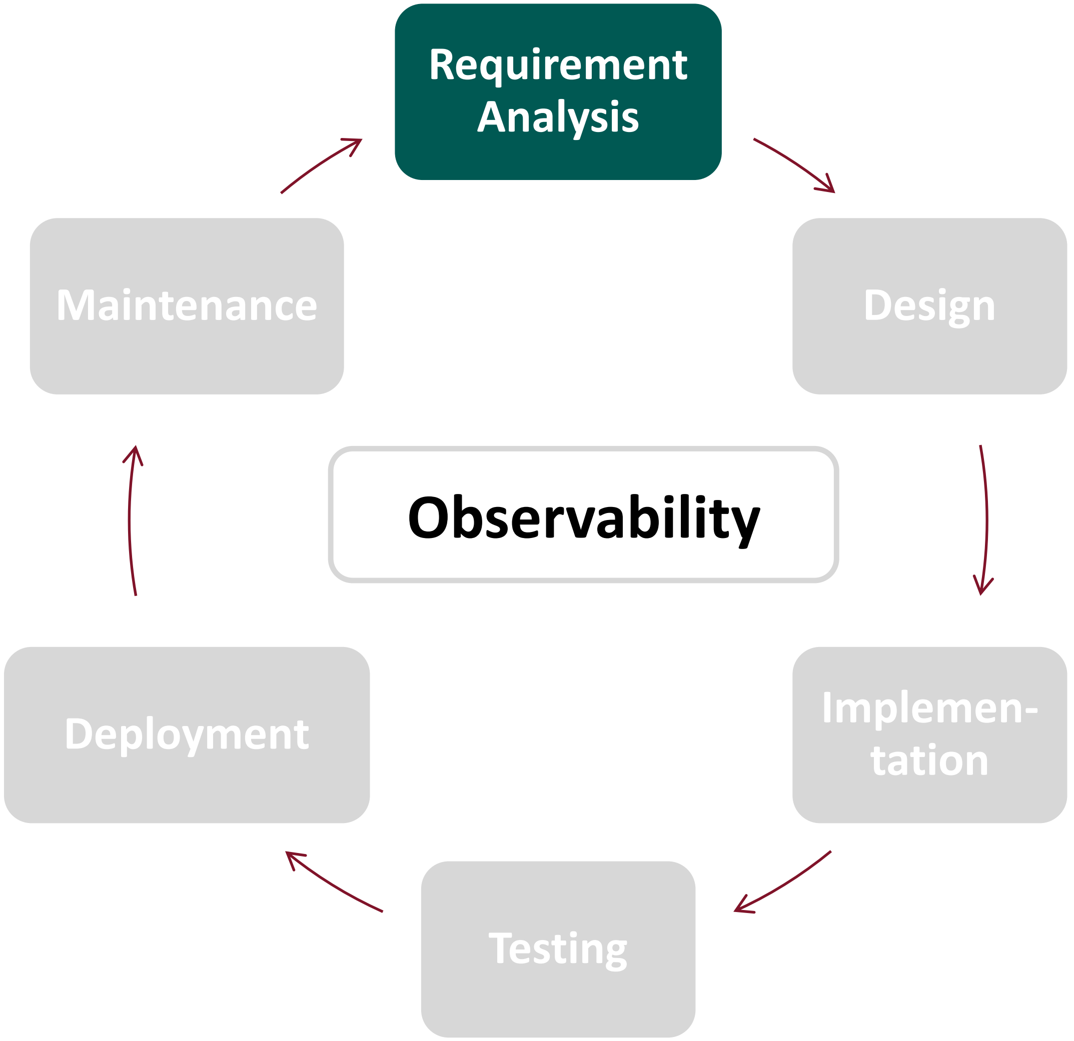
Brining observability to early stages of the SDLC using Sys Eng.

- Bringing observability **<u>to early stages</u>** of the development lifecycle
- **<u>Cost of observability</u>** can be assessed during project planning



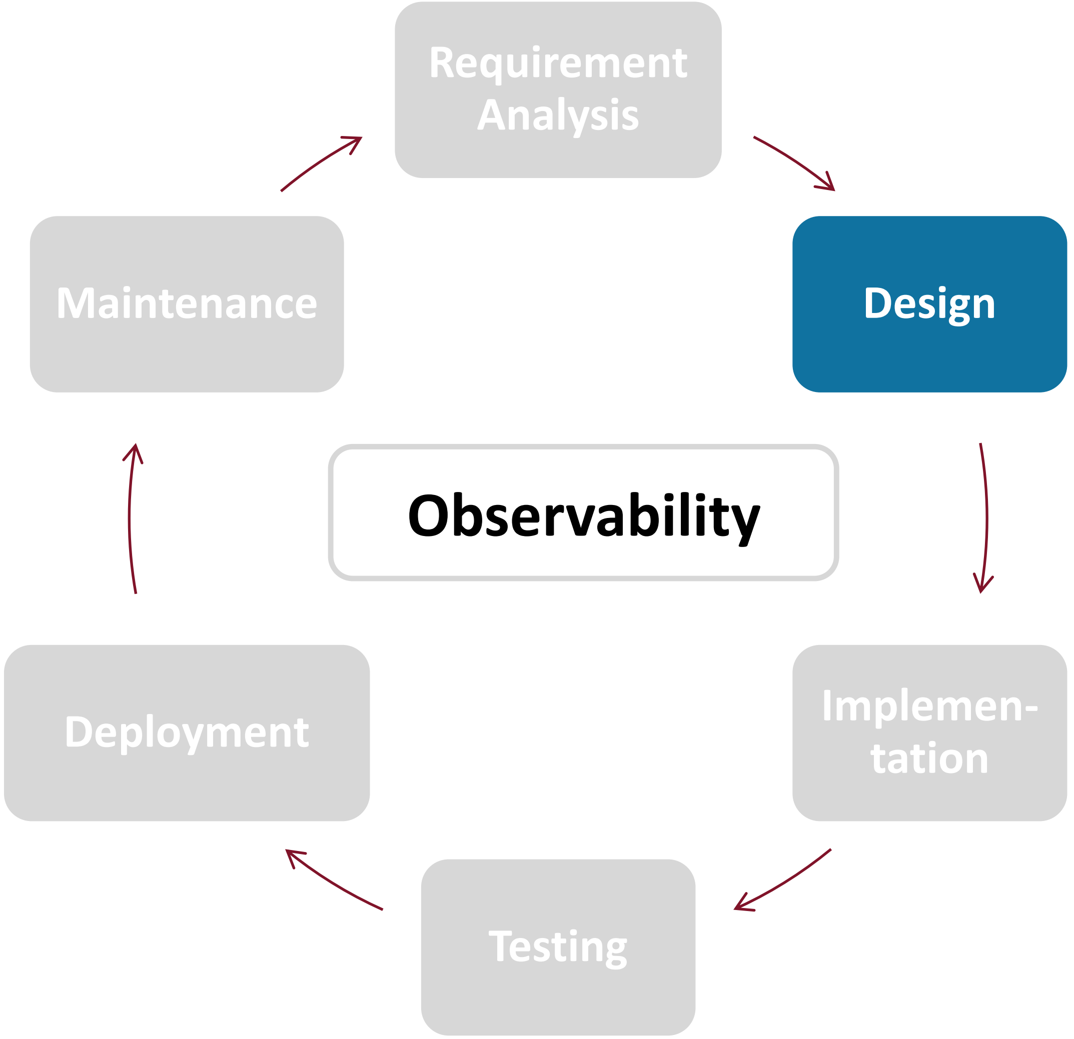
Brining observability to early stages of the SDLC using Sys Eng.

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



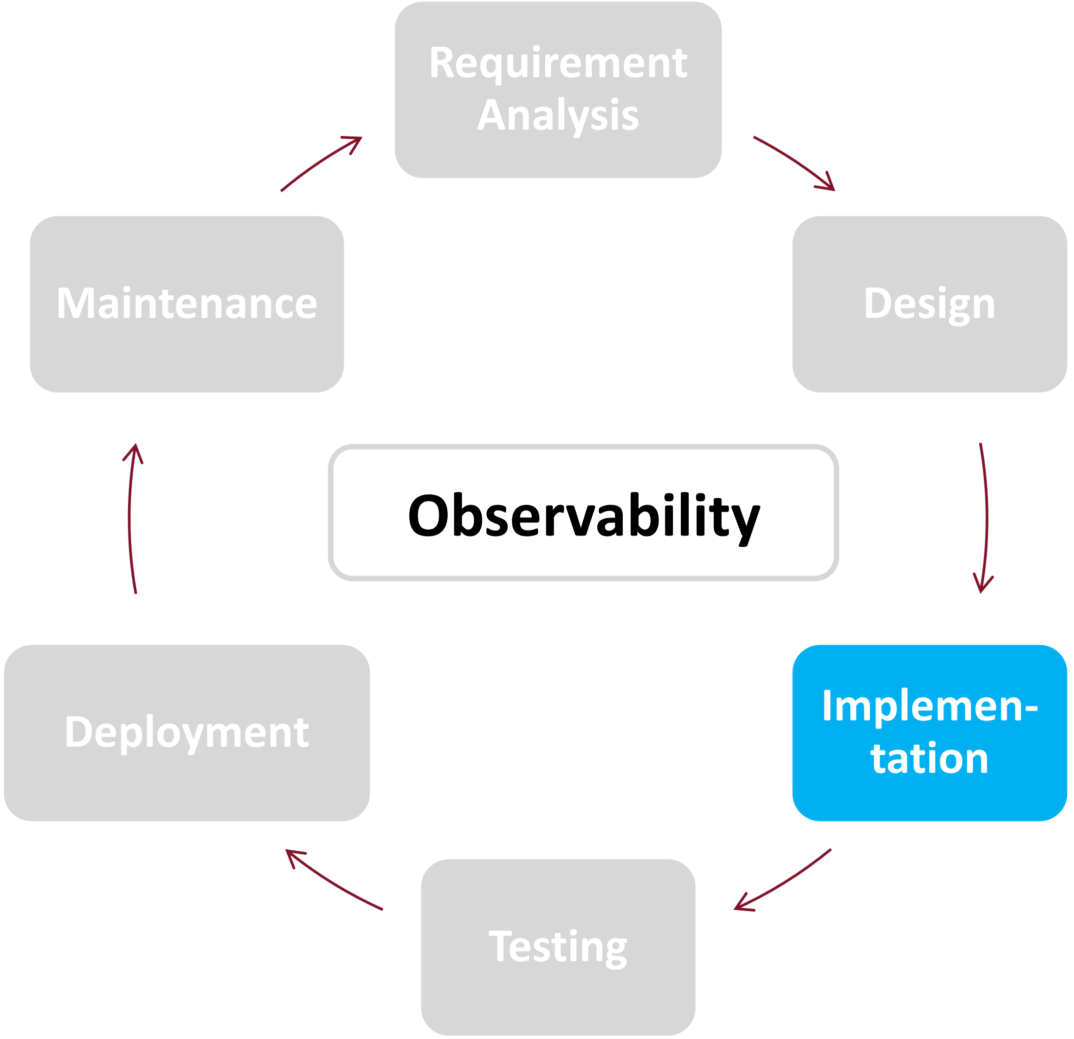
Brining observability to early stages of the SDLC using Sys Eng.

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



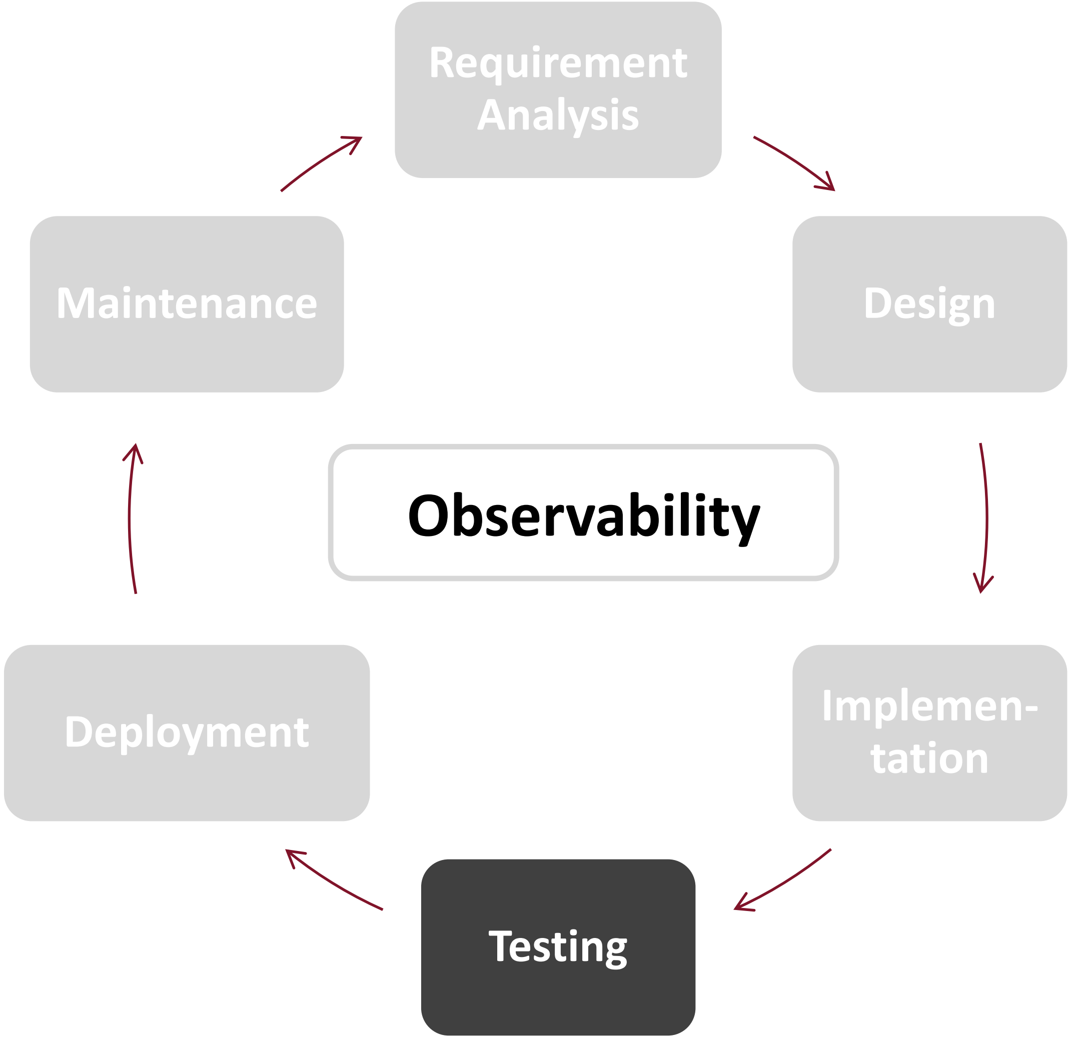
Brining observability to early stages of the SDLC using Sys Eng.

- What, where, and how to log and/or trace?
- Use of libraries and frameworks
- Patterns and best practices



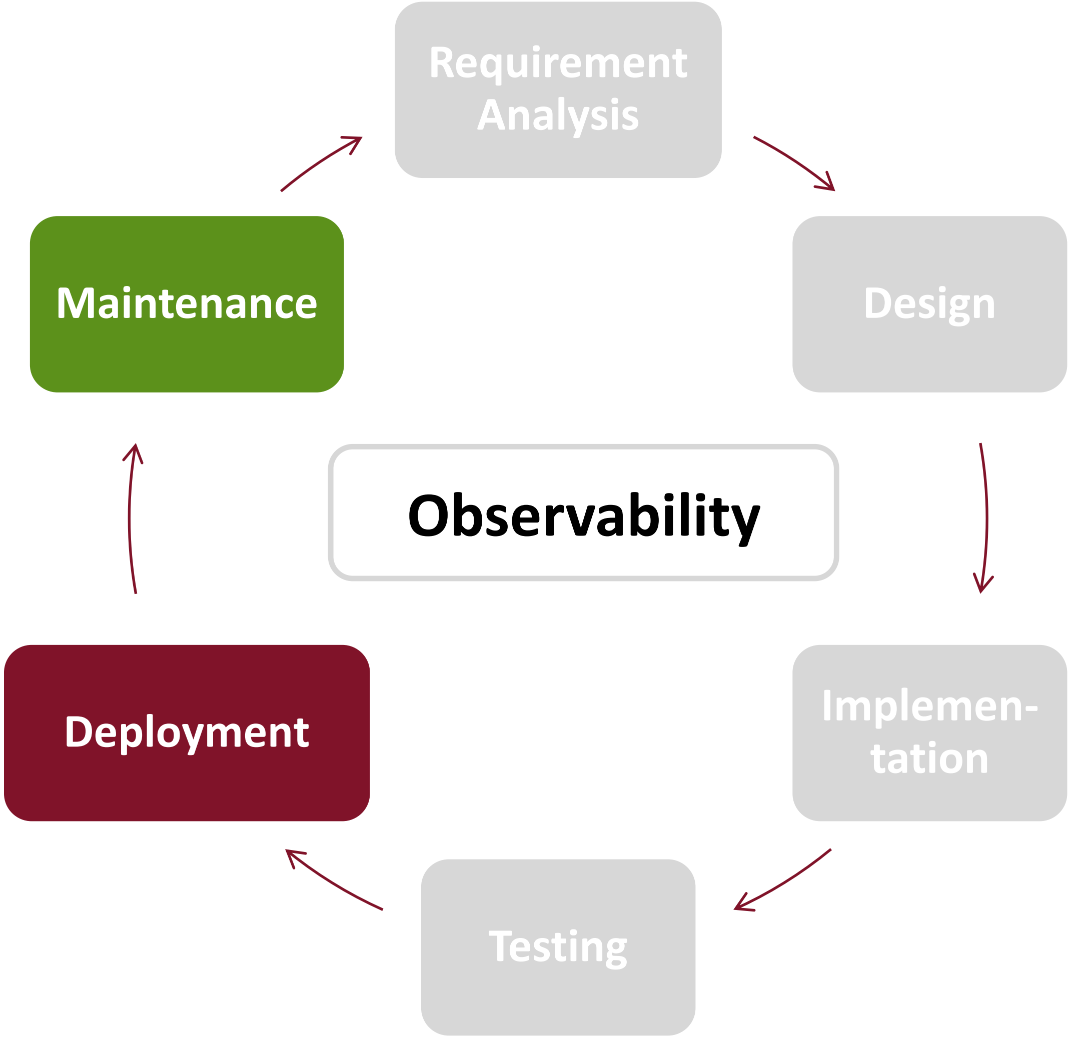
Brining observability to early stages of the SDLC using Sys Eng.

- Testing and inspection strategies for logging/tracing code



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-thejourney-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
3. https://www.nytimes.com/2021/10/12/business/economy/workers-quitting-august.html