AIware Observability

Ben Rombaut

Huawei Canada

How to cite this session?

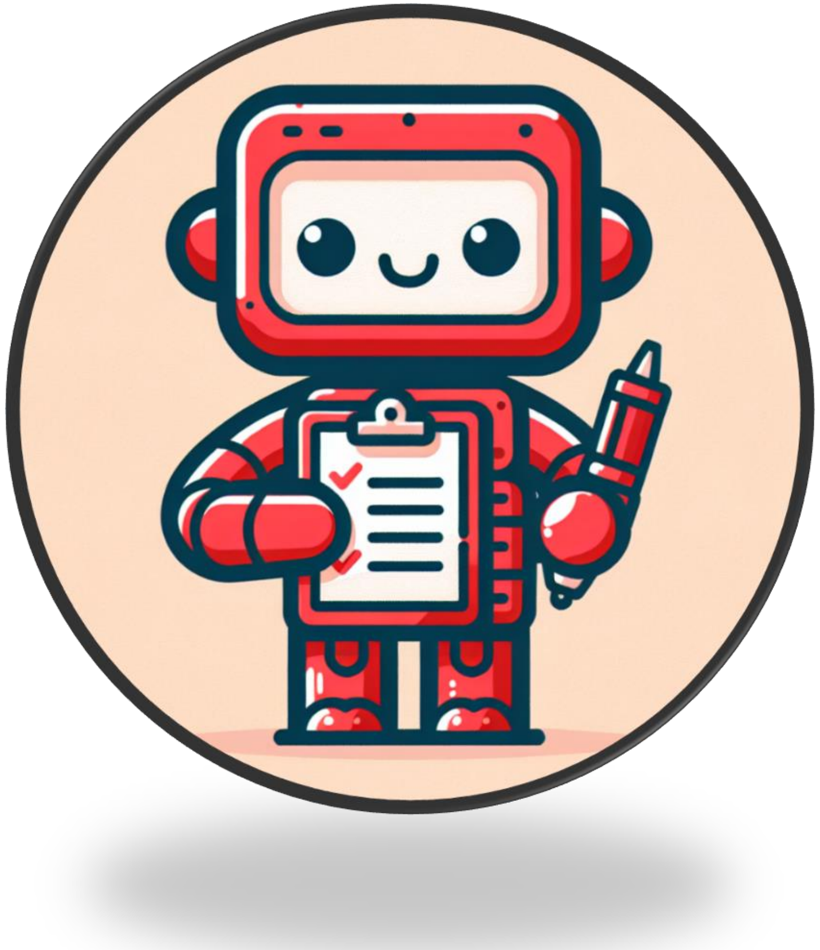
```
@misc{Rombaut2024AIwareTutorial,
author = {Benjamin Rombaut and Sogol Masoumzadeh and Kirill Vasilevski and Dayi Lin and Ahmed E. Hassan},
title = {AIware Observability},
howpublished = {Tutorial presented at the AIware Leadership Bootcamp 2024},
month = {November},
year = {2024},
address = {Toronto, Canada},
note = {Part of the AIware Leadership Bootcamp series.},
url = {https://aiwarebootcamp.io/slides/2024_aiwarebootcamp_rombaut_aiware_observability.pdf}
}
```

Check this paper for more information about this session

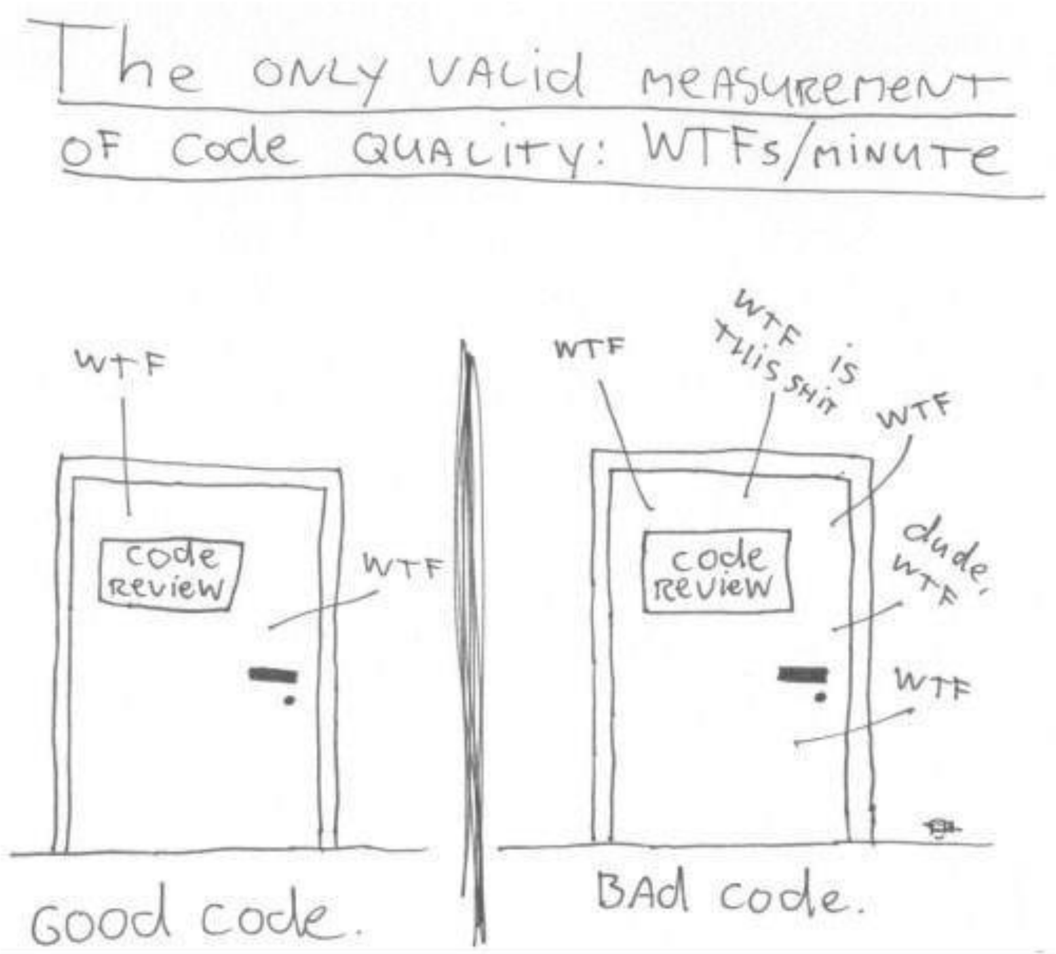
```
@article{rombaut2024watson,
  title={Watson: A Cognitive Observability Framework for the Reasoning of Foundation Model-Powered Agents},
  author={Benjamin Rombaut and Sogol Masoumzadeh and Kirill Vasilevski and Dayi Lin and Ahmed E. Hassan},
  journal={arXiv preprint arXiv:2411.03455},
  year={2024}
}
```

Overview of the session

- ☐ What is Observability?
- ☐ Operational Observability
- ☐ AIware Observability Challenges
- ☐ Cognitive Observability
- ☐ Watson: A Framework for Reasoning Path Observability

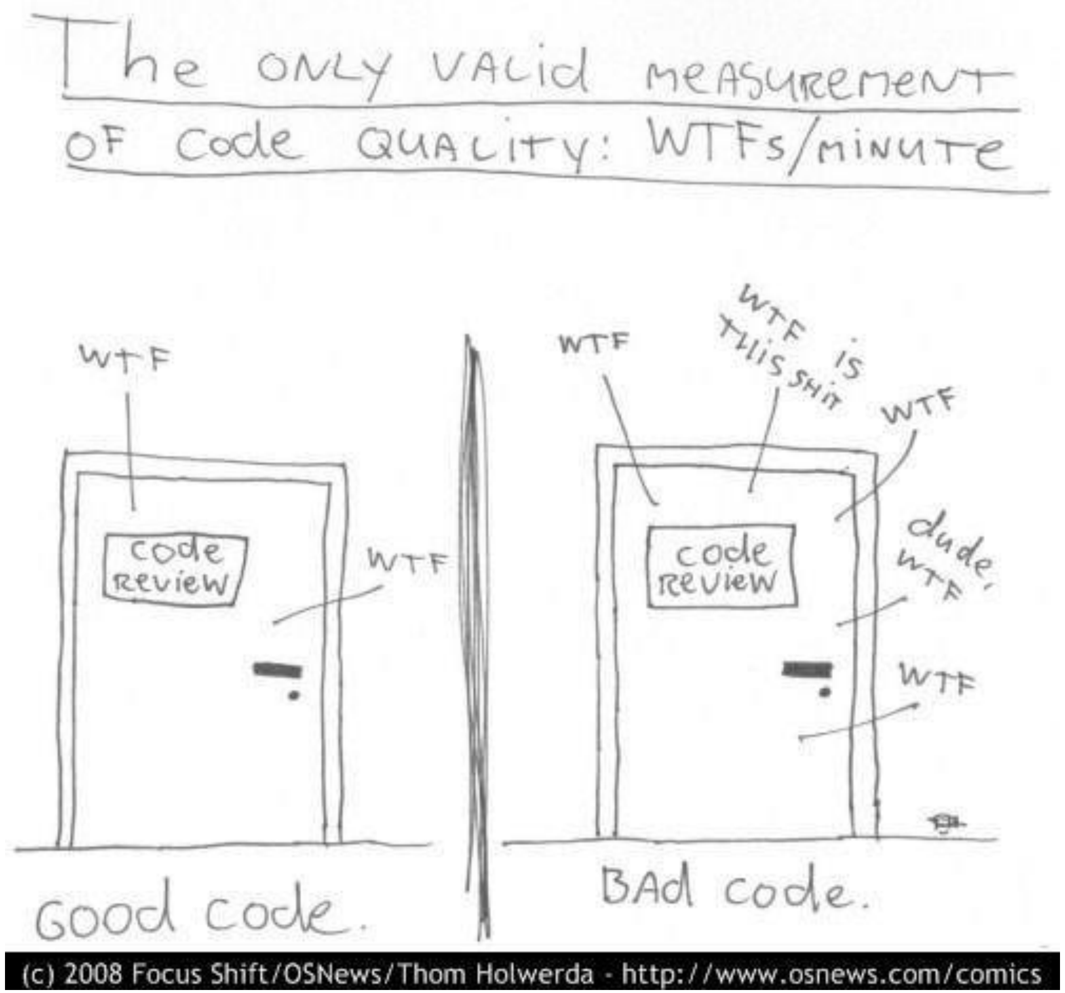


Then…

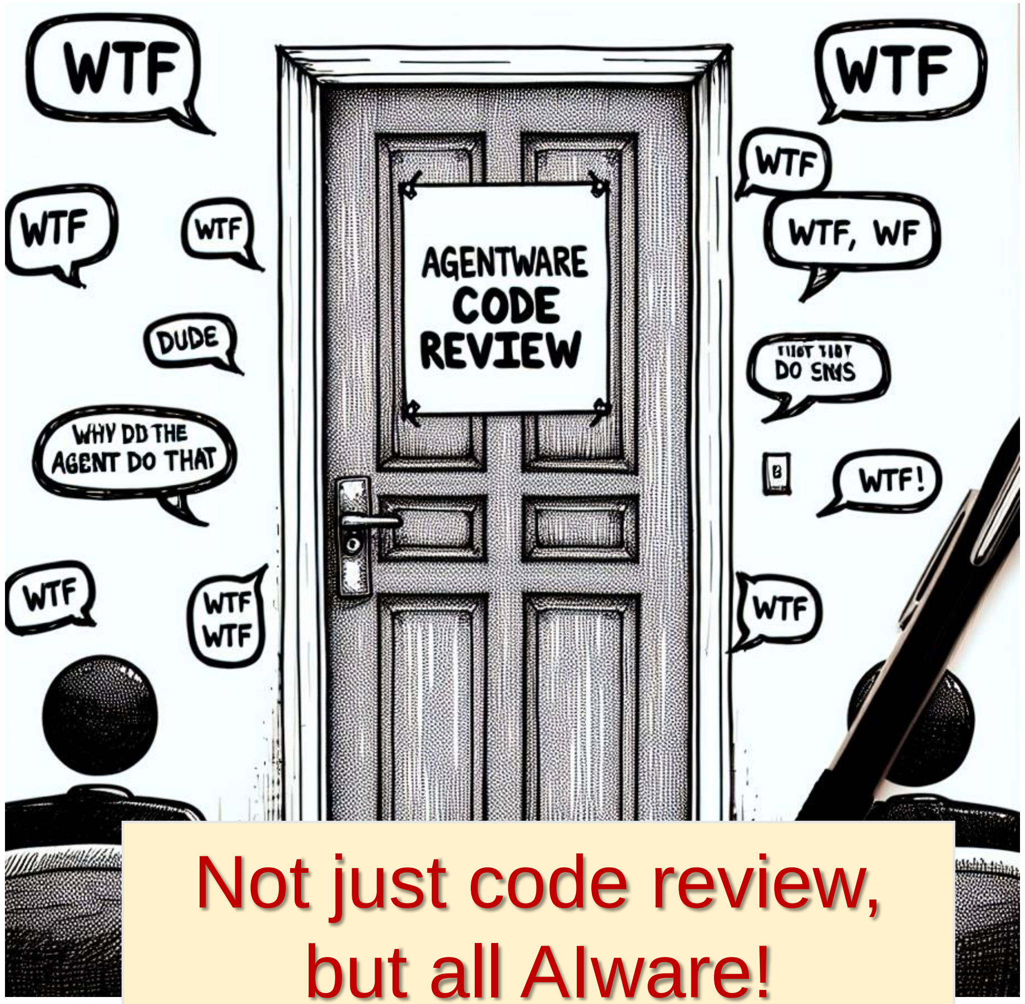


(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

Then…


The only valid measurement of code quality: WTFs/minute
WTF
code review
WTF
Good code.
WTF
WTF is this shit
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

…and Now



Not just code review,
but all AIware!

Overview of the session

- ➢ What is Observability?
- ☐ Operational Observability
- ☐ AIware Observability Challenges
- ☐ Cognitive Observability
- ☐ Watson: A Framework for Reasoning Path Observability

Observability: The ability to understand a system’s capabilities and behaviours by analyzing the data it produces

I’ve deployed my application, now…

What is happening?

Why is it happening?

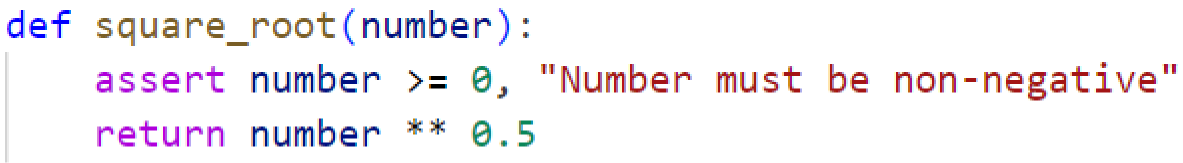
Where are potential problems?

How is the system performing?

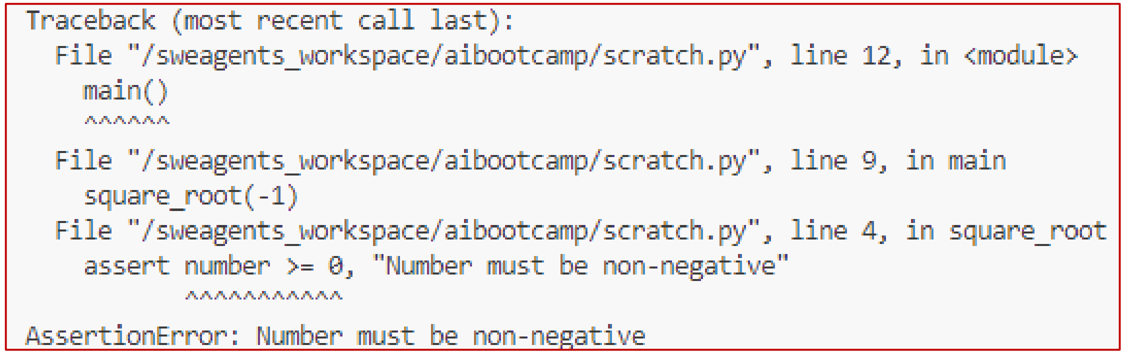
Observability in Traditional Software

Systems are instrumented to capture outputs and state

Use assertions to verify program properties.

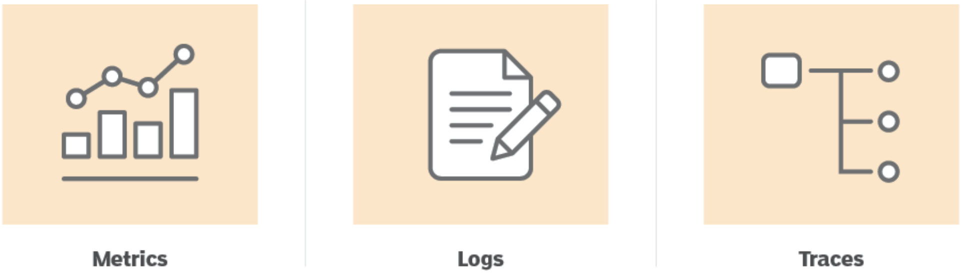


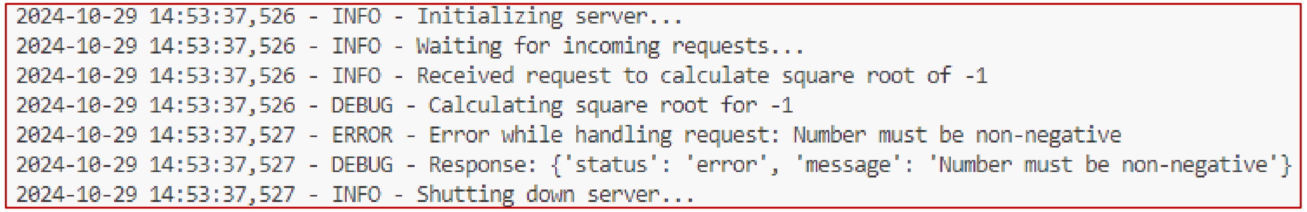
```python
def square_root(number):
    assert number >= 0, "Number must be non-negative"
    return number ** 0.5
```



```
Traceback (most recent call last):
  File "/sweagents_workspace/aibootcamp/scratch.py", line 12, in <module>
    main()
    ^^^^^^
  File "/sweagents_workspace/aibootcamp/scratch.py", line 9, in main
    square_root(-1)
  File "/sweagents_workspace/aibootcamp/scratch.py", line 4, in square_root
    assert number >= 0, "Number must be non-negative"
           ^^^^^^^^^^^
AssertionError: Number must be non-negative
```

Use metrics, logs and traces that capture the system’s state.



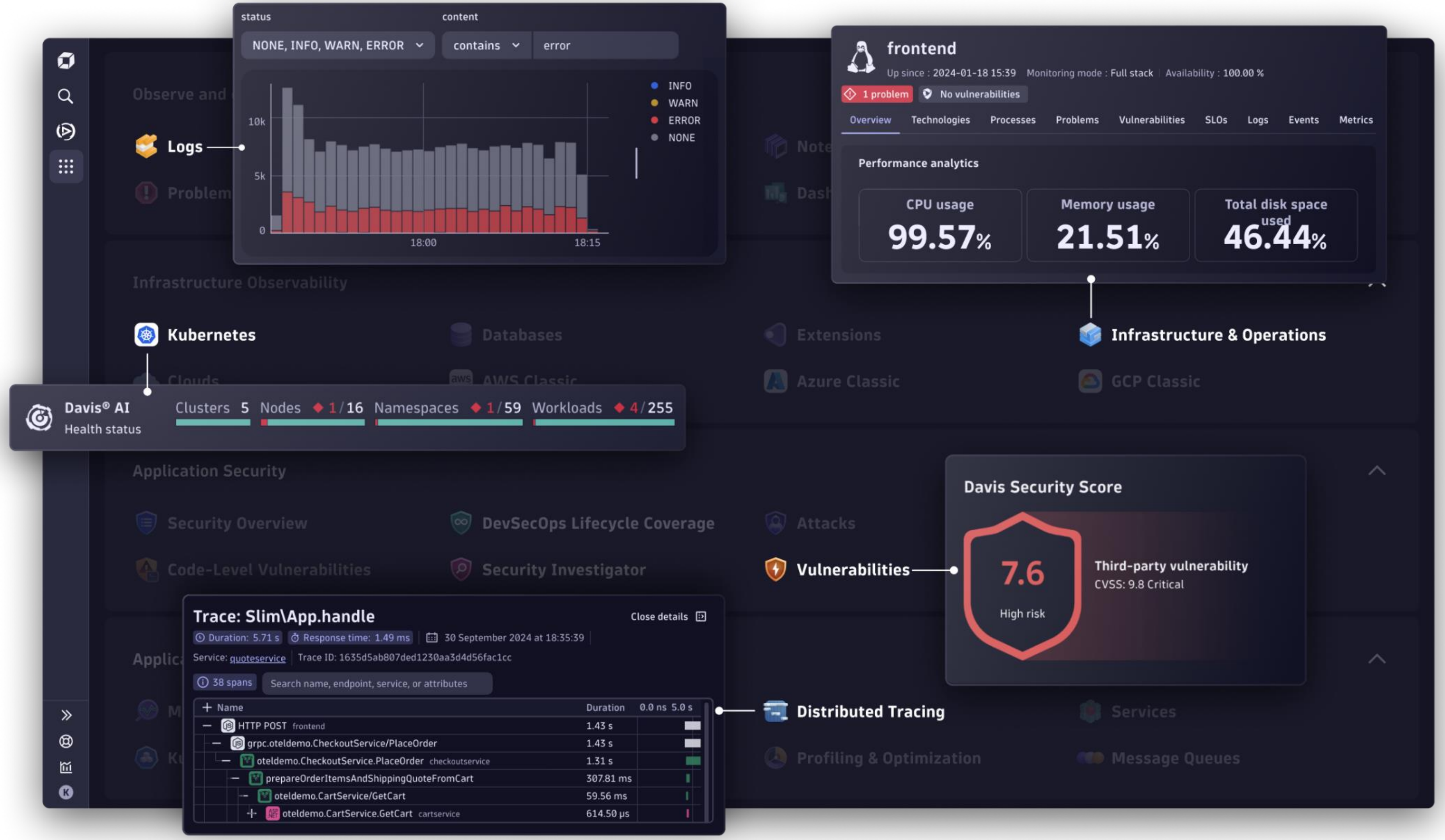


```
2024-10-29 14:53:37,526 - INFO - Initializing server...
2024-10-29 14:53:37,526 - INFO - Waiting for incoming requests...
2024-10-29 14:53:37,526 - INFO - Received request to calculate square root of -1
2024-10-29 14:53:37,526 - DEBUG - Calculating square root for -1
2024-10-29 14:53:37,527 - ERROR - Error while handling request: Number must be non-negative
2024-10-29 14:53:37,527 - DEBUG - Response: {'status': 'error', 'message': 'Number must be non-negative'}
2024-10-29 14:53:37,527 - INFO - Shutting down server...
```

- These methods are effective due to the direct traceability between the system's behavior and the executed assets.
- Developers can instrument these development assets to track execution flows, identify bottlenecks, detect anomalies, and trace issues back to their source for resolution.

Observability in Traditional Software

Observability platforms are used to track and monitor systems in production



Trace: Slim\App.handle

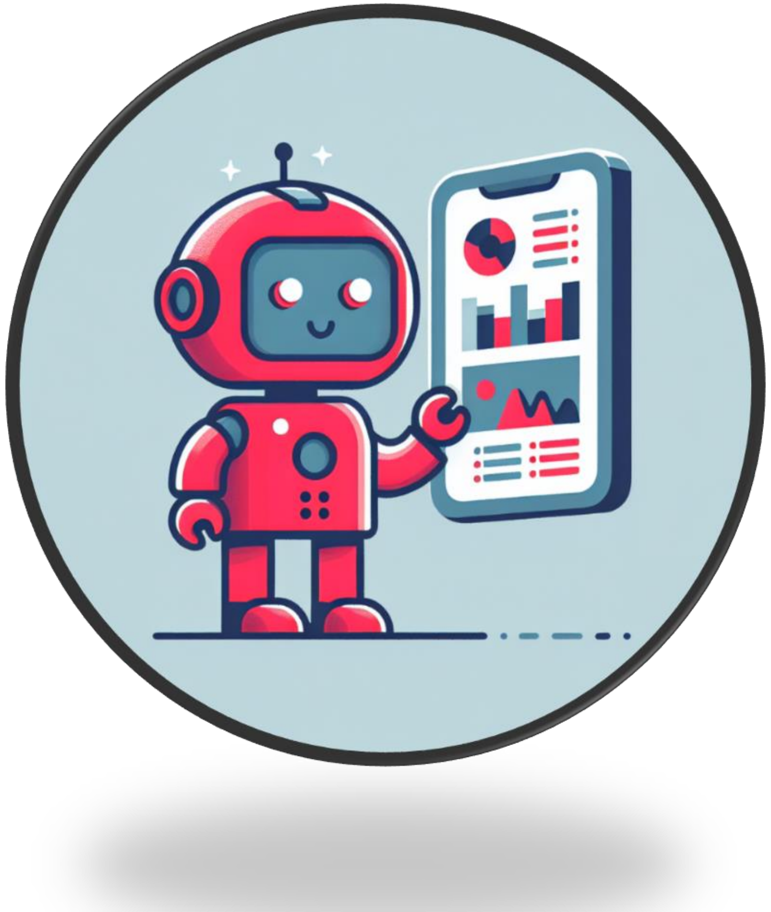
Duration: 5.71 s | Response time: 1.49 ms | 30 September 2024 at 18:35:39

Service: quoteservice | Trace ID: 1635d5ab807ded1230aa3d4d56fac1cc

Name	Duration
HTTP POST frontend	1.43 s
grpc.oteldemo.CheckoutService/PlaceOrder	1.43 s
oteldemo.CheckoutService.PlaceOrder checkoutservice	1.31 s
prepareOrderItemsAndShippingQuoteFromCart	307.81 ms
oteldemo.CartService/GetCart	59.56 ms
oteldemo.CartService.GetCart cartservice	614.50 μs

Dynatrace https://www.dynatrace.com/

Overview of the session



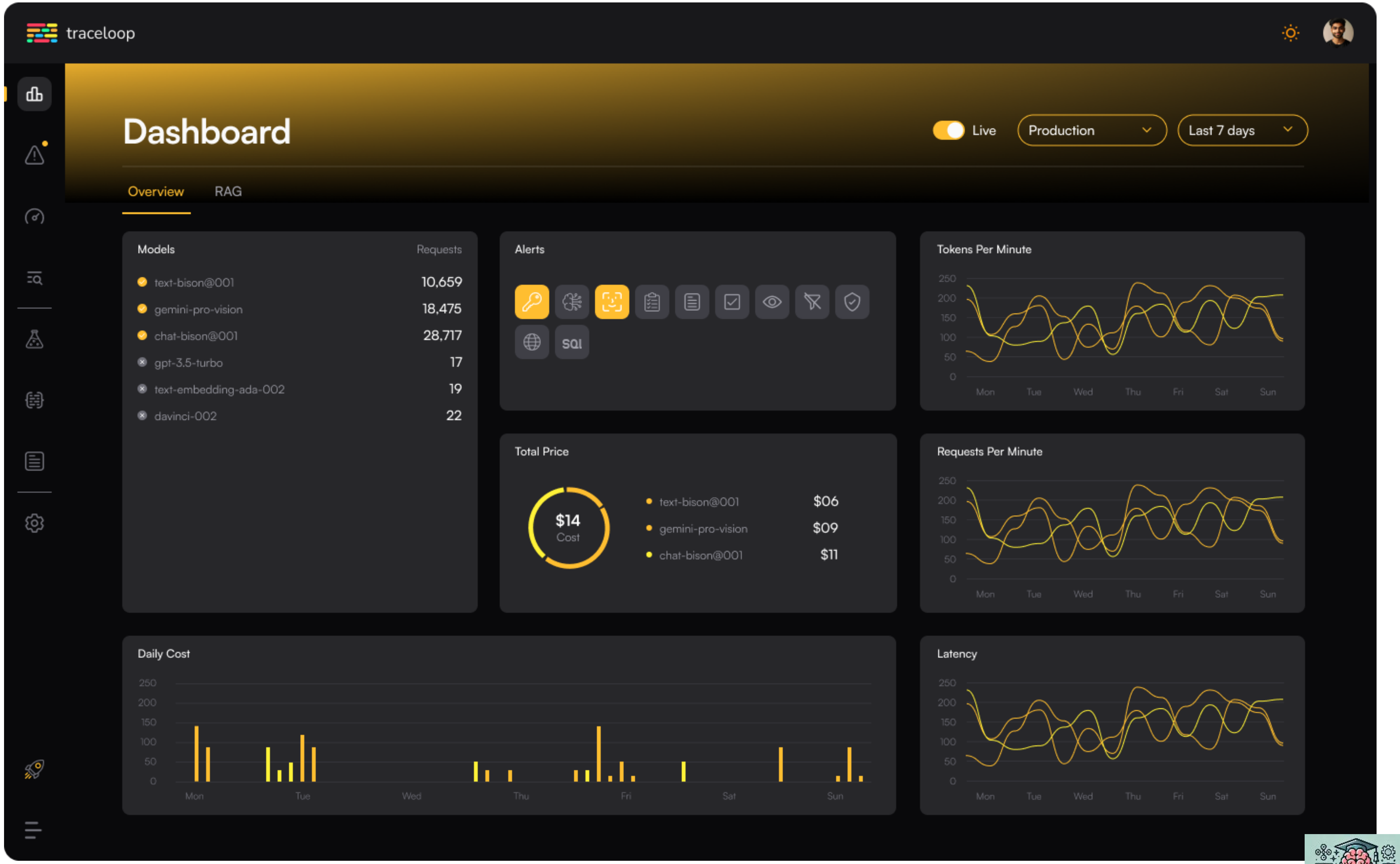
- ✓ What is Observability?
- ➢ Operational Observability
- ❑ AIware Observability Challenges
- ❑ Cognitive Observability
- ❑ Watson: A Framework for Reasoning Path Observability

Operational Observability

Observing what happened - a good start for AIware Observability

Monitoring metrics and system status through logs, traces, and performance counters.

Provides a comprehensive view of interactions between systems and components, especially in complex workflows integrating multiple model inferences and legacy system calls.



Models	Requests
text-bison@001	10,659
gemini-pro-vision	18,475
chat-bison@001	28,717
gpt-3.5-turbo	17
text-embedding-ada-002	19
davinci-002	22

Total Price

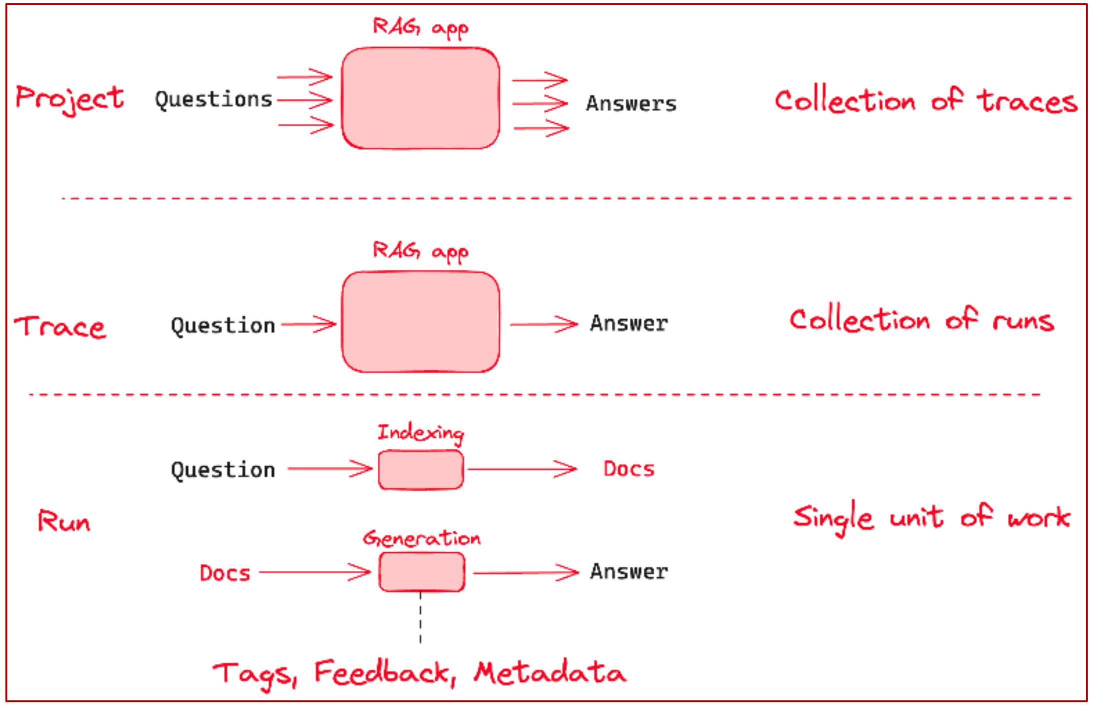
$14 Cost

text-bison@001	$06
gemini-pro-vision	$09
chat-bison@001	$11

Operational Observability

Logs & Execution Traces

Even when treating FM systems as "black boxes," we can still achieve observability by analyzing logs and execution traces, which reveal data flow between components and help detect anomalies in execution paths.



Going beyond plain API monitoring

Extends traditional API monitoring to include FM interactions.

Covers more than just requests and errors (e.g., prompts, responses, costs, and token usage).

Deeper insights into prompt evaluation and model performance.

Focuses on FM functionality and performance within applications.

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Operational Observability

Metrics, Resources, & LLM Tool Monitoring

Volume Metrics	Latency Metrics	Token Metrics	Resource Utilization	Tool Monitoring
Allow us to track the number of calls to specific FMs in AIware.	Provide measurements of response delays across AIware, aiding in the identification of potential bottlenecks and the enhancement of the user experience.	Offer insights into model behavior, revealing patterns such as prompt and completion usage that can impact overall performance and cost	Ensures that compute, memory, and storage capacities are efficiently used by FMs.	Understand how AIware components interact, enabling an assessment of their capabilities and effectiveness within the AIware architecture.

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

State-of-the-Art Observability Tools & Platforms

Framework	Trace or Request-Response (RR) Monitoring	Volume		Latency (PX)						Tokens			Resources	
		Trace Count/Status	LLM Call Count/Status	Trace Latency	LLM Latency	LLM Calls per Trace	Tokens / sec	Trace Time-to-First-Token	LLM Time-to-First-Token	Total Tokens	Tokens per Trace	Tokens per LLM Call	Cost	HW Util.
LangSmith	Trace	Yes		Yes						Yes			Yes	
HumanLoop	RR	Yes		Yes						No			No	
Qwak	Trace	Yes		Yes						Yes			Yes	
Helicone	RR	Yes		Yes						Yes			Yes	
Langfuse	Trace	Yes		Yes						Yes			Yes	
Dynatrace	Trace	Yes		Yes (very simple)						Yes			Yes	
Pheonix (Arize)	Trace	Yes		Yes						Yes			Yes (cost – Arize)	
Lunary	Trace	Yes		Yes						Yes			Yes (cost)	
LangKit (WhyLabs)	Trace	Yes		No (?)						No (?)			No (?)	
Laminar	Trace	Yes		Yes						Yes			No	
TraceLoop	Trace	Yes		Yes						Yes			No	
DataDog	RR	Yes		Yes						Yes			Yes (cost)	

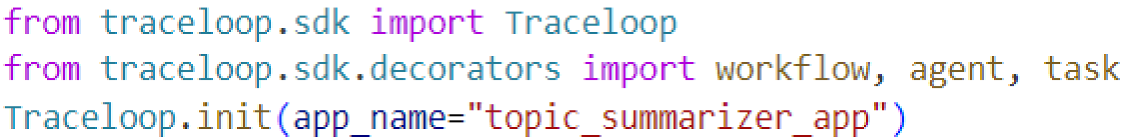
Focusing on low level details, primarily collecting traces and runs consisting of LLM calls, Vector DB calls, user prompts, and associated metrics (e.g., volume, latency, token counts, resource utilization, etc.).

OpenLLMetry https://github.com/traceloop/openllmetry

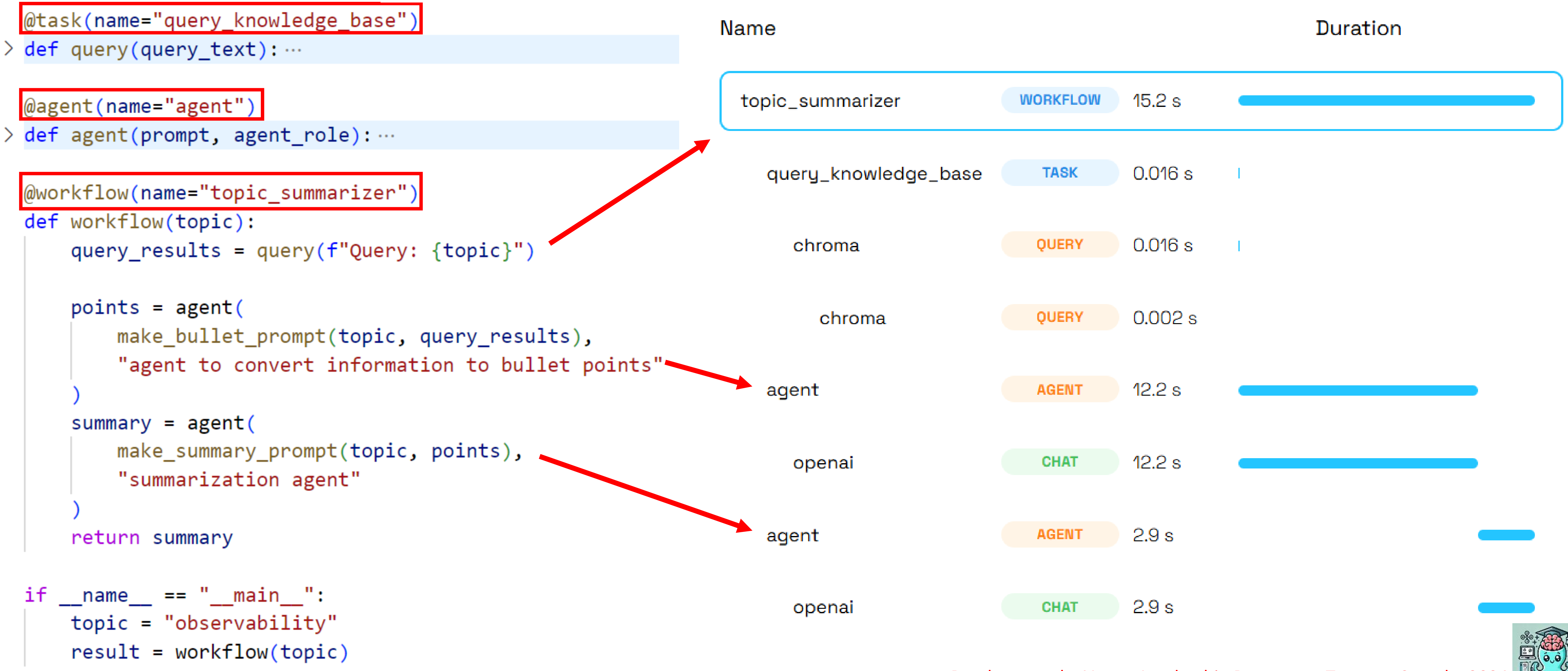
- Use existing standard OpenTelemetry instrumentations for LLM providers and Vector DBs
- Support some new LLM-specific extensions for example OpenAI, Anthropic API calls

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

State-of-the-Art Observability Tools & Platforms



```
from traceloop.sdk import Traceloop
from traceloop.sdk.decorators import workflow, agent, task
Traceloop.init(app_name="topic_summarizer_app")
```

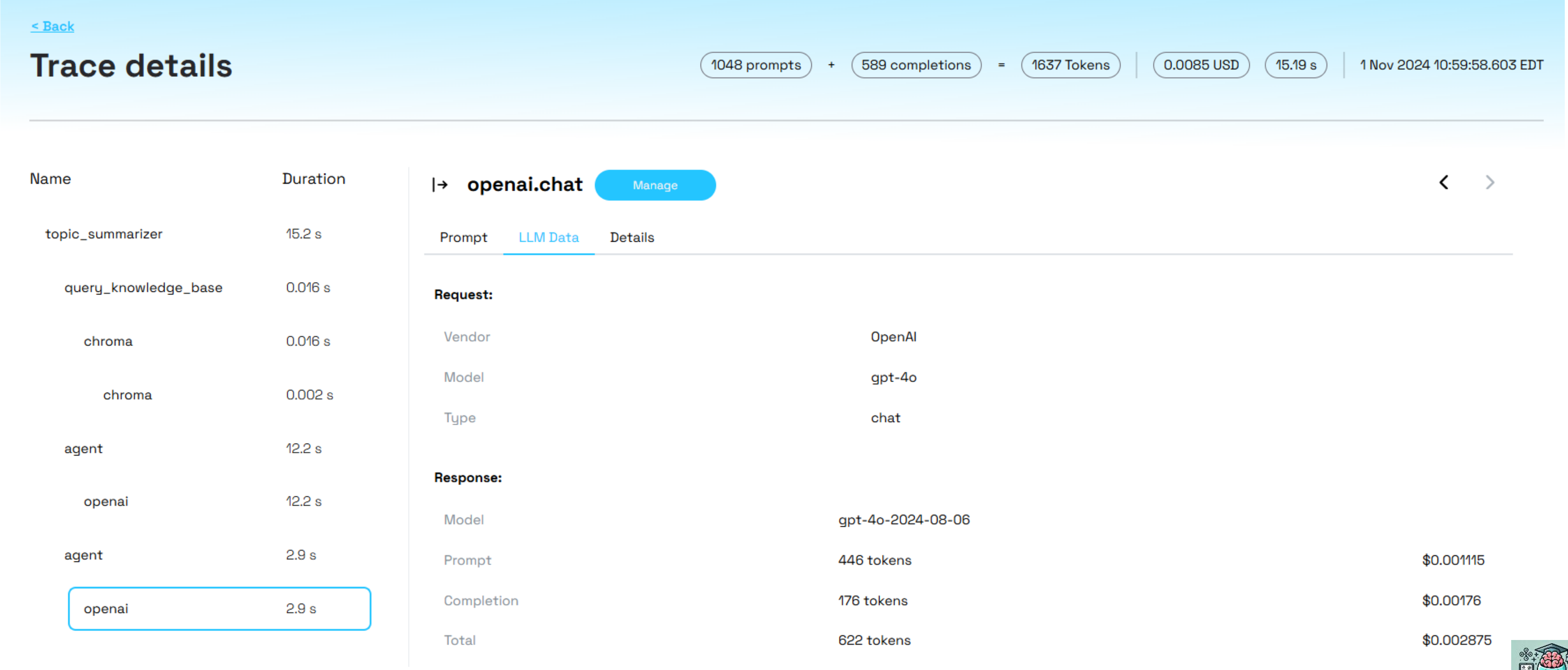


```
@task(name="query_knowledge_base")
> def query(query_text): ⋯

@agent(name="agent")
> def agent(prompt, agent_role): ⋯

@workflow(name="topic_summarizer")
def workflow(topic):
    query_results = query(f"Query: {topic}")

    points = agent(
        make_bullet_prompt(topic, query_results),
        "agent to convert information to bullet points"
    )
    summary = agent(
        make_summary_prompt(topic, points),
        "summarization agent"
    )
    return summary

if __name__ == "__main__":
    topic = "observability"
    result = workflow(topic)
```

Name		Duration
topic_summarizer	WORKFLOW	15.2 s
query_knowledge_base	TASK	0.016 s
chroma	QUERY	0.016 s
chroma	QUERY	0.002 s
agent	AGENT	12.2 s
openai	CHAT	12.2 s
agent	AGENT	2.9 s
openai	CHAT	2.9 s

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

State-of-the-Art Observability Tools & Platforms



Trace details

1048 prompts + 589 completions = 1637 Tokens | 0.0085 USD | 15.19 s | 1 Nov 2024 10:59:58.603 EDT

Name	Duration
topic_summarizer	15.2 s
query_knowledge_base	0.016 s
chroma	0.016 s
chroma	0.002 s
agent	12.2 s
openai	12.2 s
agent	2.9 s
openai	2.9 s

openai.chat

Request:

Vendor	OpenAI
Model	gpt-4o
Type	chat

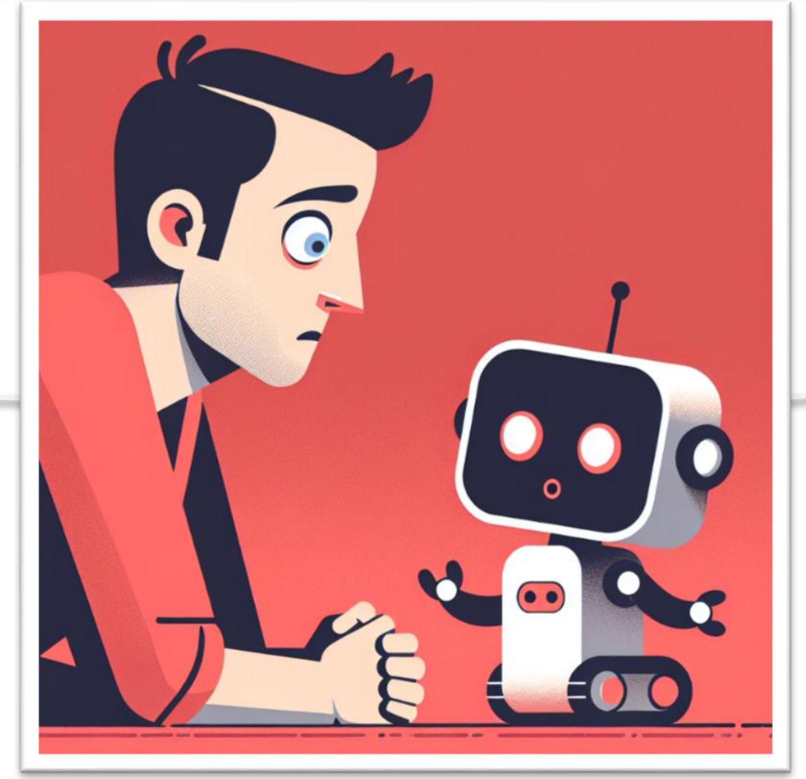
Response:

Model	gpt-4o-2024-08-06	
Prompt	446 tokens	$0.001115
Completion	176 tokens	$0.00176
Total	622 tokens	$0.002875

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

However, as FMs become more capable and AIware becomes more complex, the requirements are shifting to higher levels of abstraction.

Which knowledge did the FM agent use in its reasoning when planning the execution of this workflow?



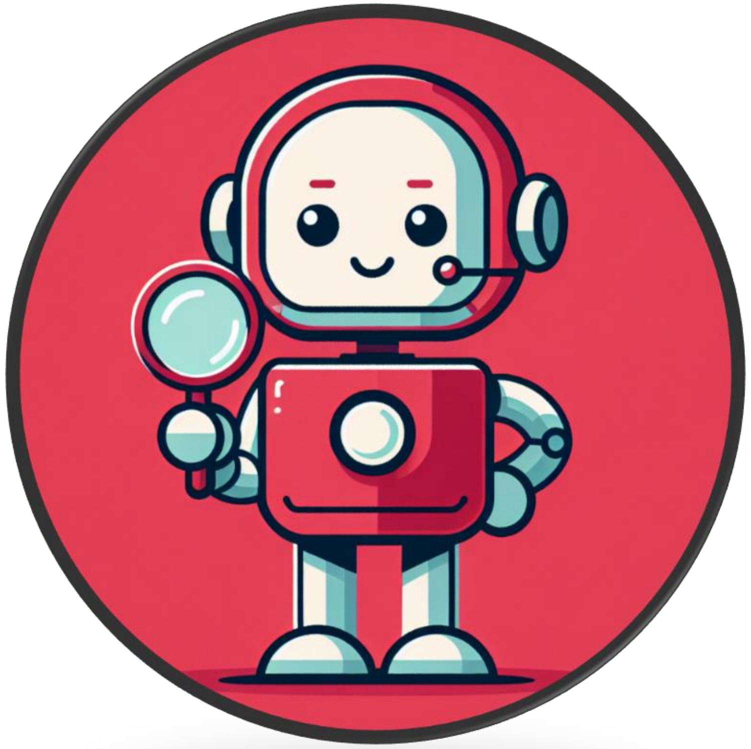
What lead the group of collaborating agents to get stuck in a loop, without reaching a solution?

But achieving this level of observability is hard!

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Overview of the session

- ✓ What is Observability?
- ✓ Operational Observability
- ➢ AIware Observability Challenges
- ☐ Cognitive Observability
- ☐ Watson: A Framework for Reasoning Path Observability



For AIware, observability goes beyond traditional monitoring

- It involves gaining deep insights into the system’s complex behaviors and semantic outputs, not just its inputs/outputs or logs.
- Achieving this level of observability is challenging because AIware's unique functionality and outputs require more advanced tools and approaches than those used for standard software monitoring.

These challenges arise from Agents because of…

Abstract Behaviors

Reasoning Chains

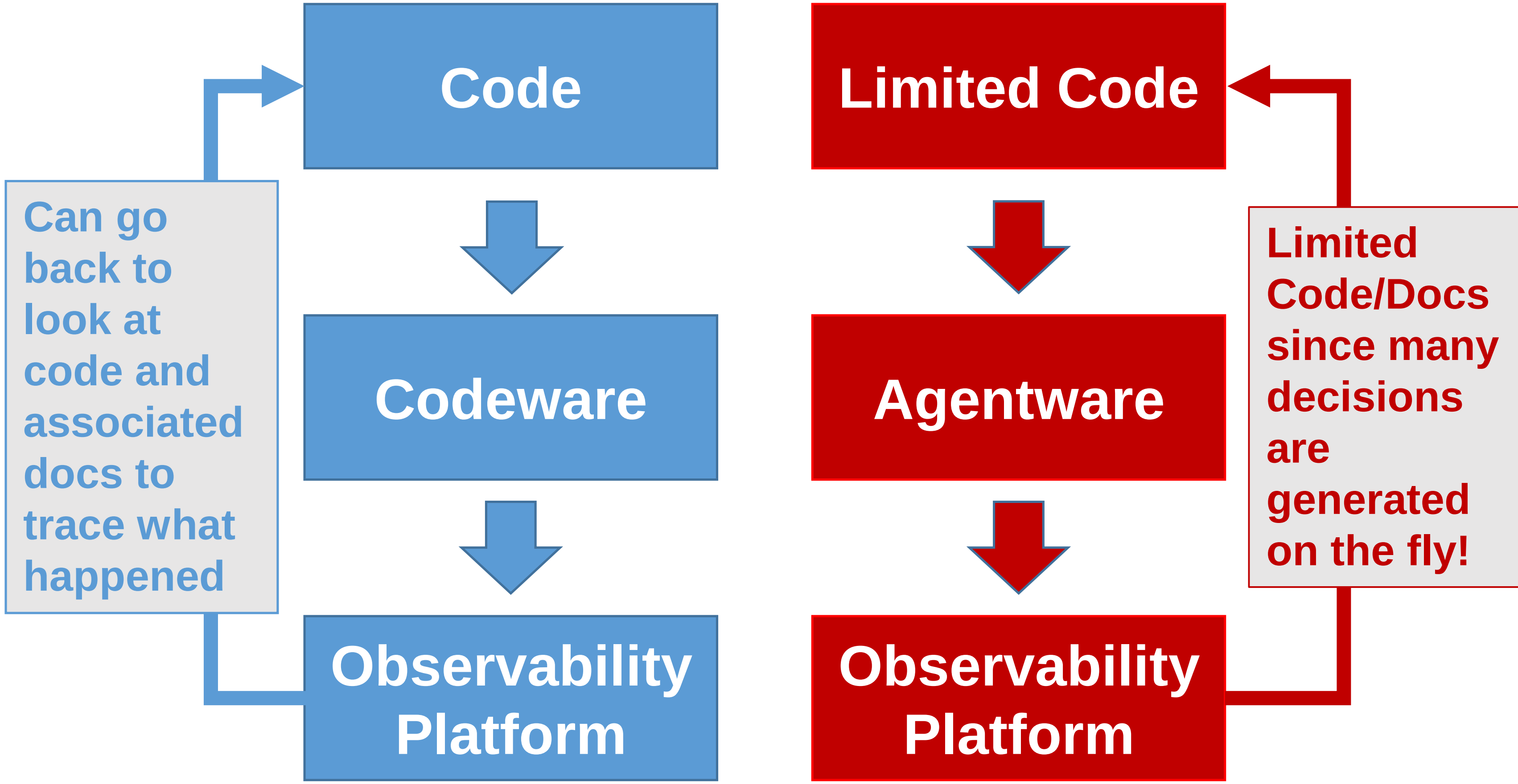
Cascading Actions

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

For AIware, observability goes beyond traditional monitoring

Abstract Behaviors

In Agentware, errors stem from abstract behaviors of agents that lack explicit code, making them difficult to observe and complicating traditional debugging techniques for understanding agent behavior.



Code

Codeware

Observability Platform

Can go back to look at code and associated docs to trace what happened

Limited Code

Agentware

Observability Platform

Limited Code/Docs since many decisions are generated on the fly!

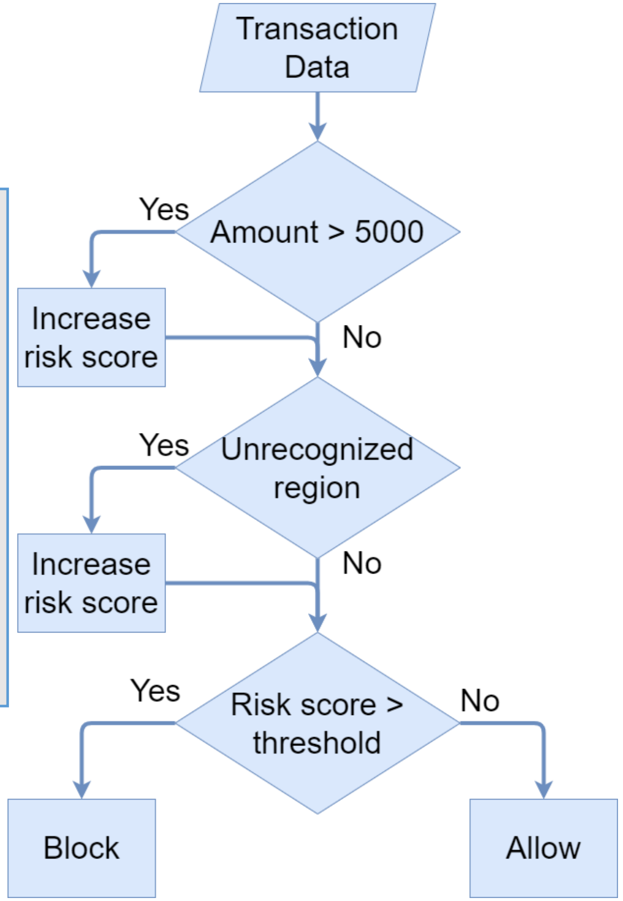
For AIware, observability goes beyond traditional monitoring

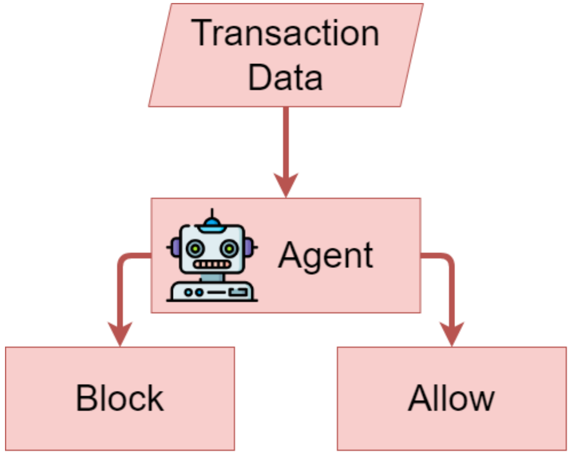
Reasoning Chains

Errors are difficult to trace as they often arise from complex reasoning chains or interactions within Agentware, with no clear logic to inspect due to decision-making being based on latent knowledge embedded in the model.

Example: Fraud Detection System

If a transaction is wrongly flagged as fraudulent, you can trace back to specific conditions and adjust the threshold or rule accordingly.





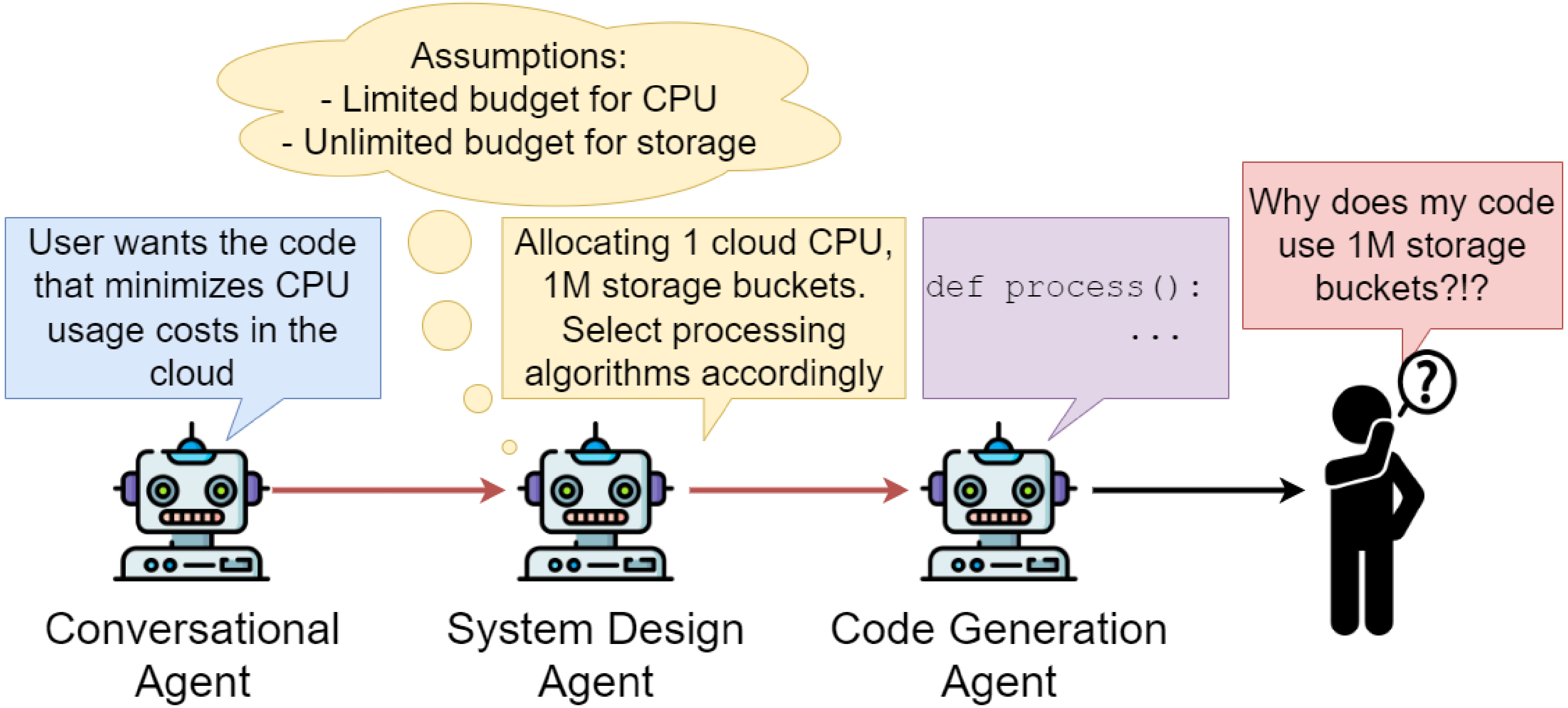
If a transaction is wrongly flagged as fraudulent, there's no clear logic path to inspect - errors could arise from complex interactions of multiple features learned by the model, and it's challenging to pinpoint exactly why the model flagged the transaction.

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

For AIware, observability goes beyond traditional monitoring

Cascading Actions

In multi-agent systems, errors from early agents can manifest later in the workflow, complicating source tracing. An incorrect assumption or action by one agent can trigger a cascade of misaligned actions by downstream agents, leading to unpredictable outcomes.

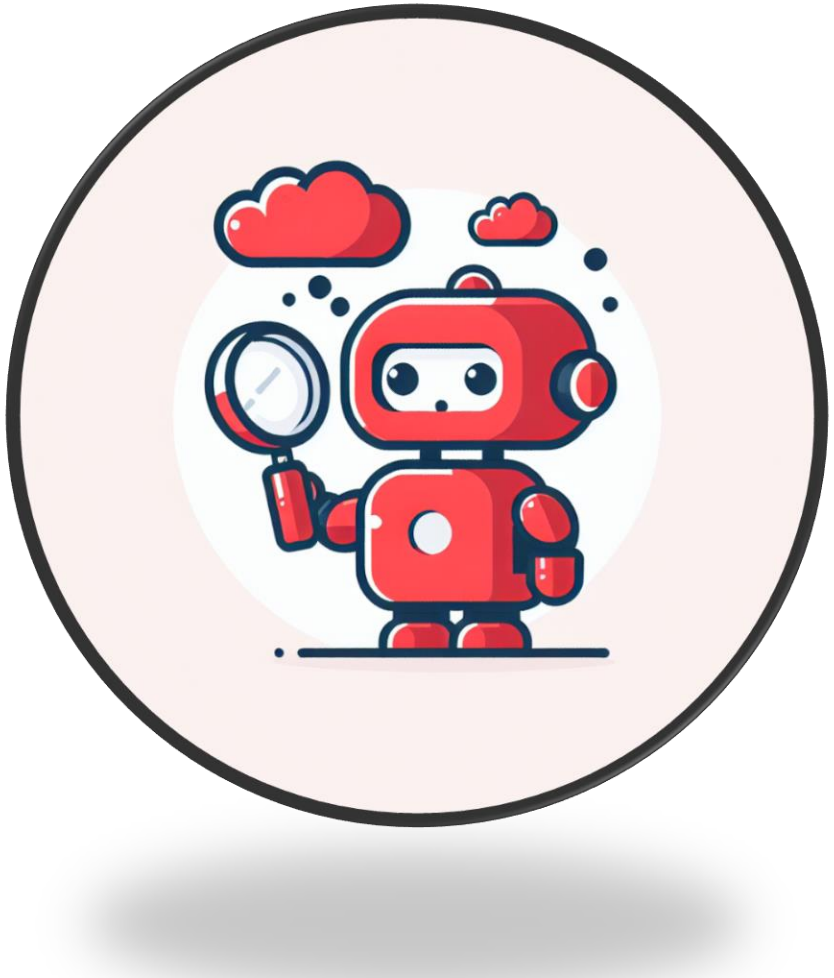

Assumptions:
- Limited budget for CPU
- Unlimited budget for storage
User wants the code that minimizes CPU usage costs in the cloud
Allocating 1 cloud CPU, 1M storage buckets. Select processing algorithms accordingly
def process():
...
Why does my code use 1M storage buckets?!?
Conversational Agent
System Design Agent
Code Generation Agent

So how should we move forward?

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Overview of the session

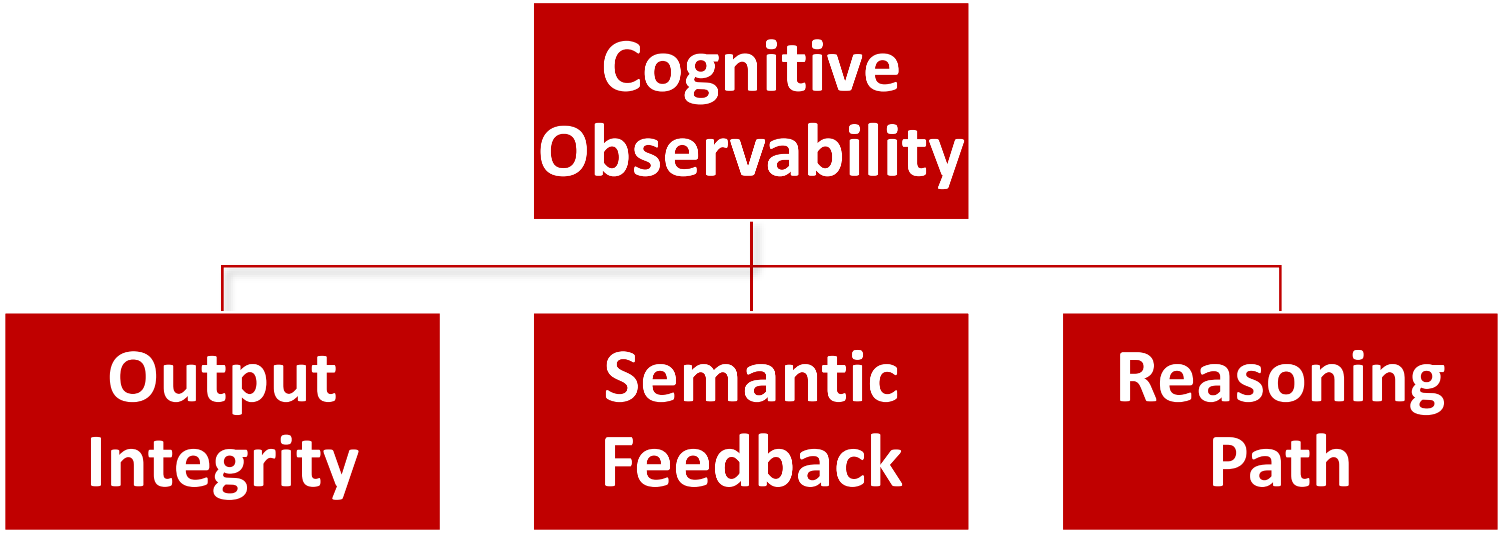
- ✓ What is Observability?
- ✓ Operational Observability
- ✓ AIware Observability Challenges
- ➢ Cognitive Observability
- ☐ Watson: A Framework for Reasoning Path Observability



Cognitive Observability

Shifts the focus from traditional technical metrics to cognitive and linguistic aspects of observability.

- Emphasizes understanding how FMware systems think and communicate, providing deeper insights into system operations.
- Provides a holistic understanding of FMware functionality beyond conventional observability metrics.
- Facilitates effective management and coordination within complex, multi-agent environments.
- Allows organizations to address sophisticated challenges, improving system trustworthiness, accountability, and transparency.
- Supports proactive intervention strategies to resolve issues before they impact end-users or business objectives.



Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Output Integrity

“How reliable and accurate are the agent’s outputs?”

Focuses on analyzing agents' outputs to gain qualitative insights into behavior.
Provides critical feedback on model reliability and coherence, and helps identify inconsistencies, biases, or errors beyond traditional metrics.

Is there a drift in the types of prompts you expect or a concept drift in how your model is responding?

Are your prompts and responses high quality (readable, understandable, well written)?

Is the LLM responding in the right tone?

Is your LLM receiving adversarial attempts or malicious prompt injections?

Are your upstream prompts changing their sentiment suddenly or over time?

Is your LLM responding with relevant content?

Are you seeing a divergence from your anticipated topics?

WhyLabs Langkit https://whylabs.ai/langkit

WHY LABS

Uses natural language techniques to extract actionable insights about prompts and responses, which can be used to identify and mitigate FM-related issues.

Metric	Description
Hallucination	Consistency between response and additional response samples
Injections	Semantic Similarity from known prompt injections and harmful behaviors
Input/Output	Semantic similarity between prompt and response
PII	Private entities identification
Proactive Injection Detection	LLM-powered proactive detection for injection attacks
Regexes	Regex pattern matching for sensitive information
Sentiment	Sentiment Analysis
Text Statistics	Text quality, readability, complexity, and grade level.
Themes	Semantic similarity between customizable groups of examples
Topics	Text classification into predefined topics - law, finance, medical, etc.
Toxicity	Toxicity, harmfulness and offensiveness

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Semantic Feedback

“What did we think of the agent’s actions?”

Explicit Feedback

- Direct user ratings of FMware-generated content.
- E.g., "Thumbs up/down" ratings in OpenAI's ChatGPT.

Implicit Feedback

- Inferred from user actions like "copied," "saved," or "dismissed.“
- E.g., GitHub Copilot's tracking of code suggestion retention and edits.

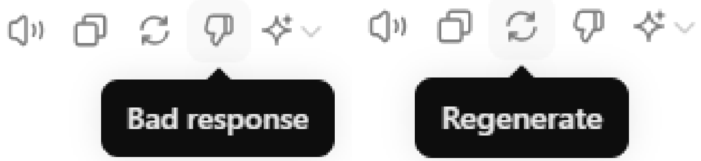
Free-form Feedback

- Corrections or explanations provided by users.

Benefits
- Gauges efficacy of FMware outputs.
- Optimizes user experiences in FMware applications.
- Informs model training and fine-tuning decisions.

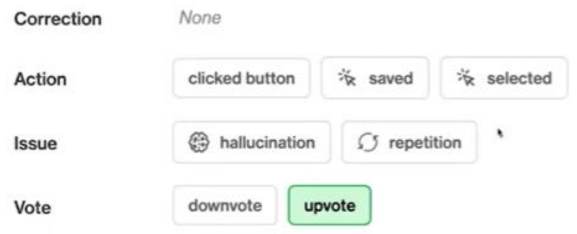
Challenges
- Complexity in interpreting feedback with increased interactions.
- Difficulty pinpointing improvement areas in multi-agent systems.
- Complex agent dynamic adaptations can obscure feedback insights.

ChatGPT



Humanloop

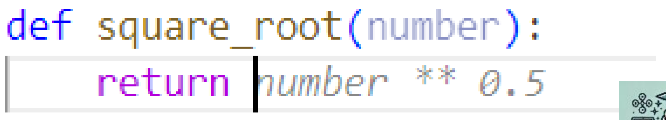
- Offers an evaluation suite to measure and improve LLM performance during development and production



Correction	None
Action	clicked button saved selected
Issue	hallucination repetition
Vote	downvote upvote

GitHub Copilot

- Tracks code suggestion acceptance and modification at different time intervals after the insertion.



```
def square_root(number):
    return number ** 0.5
```

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Reasoning Path

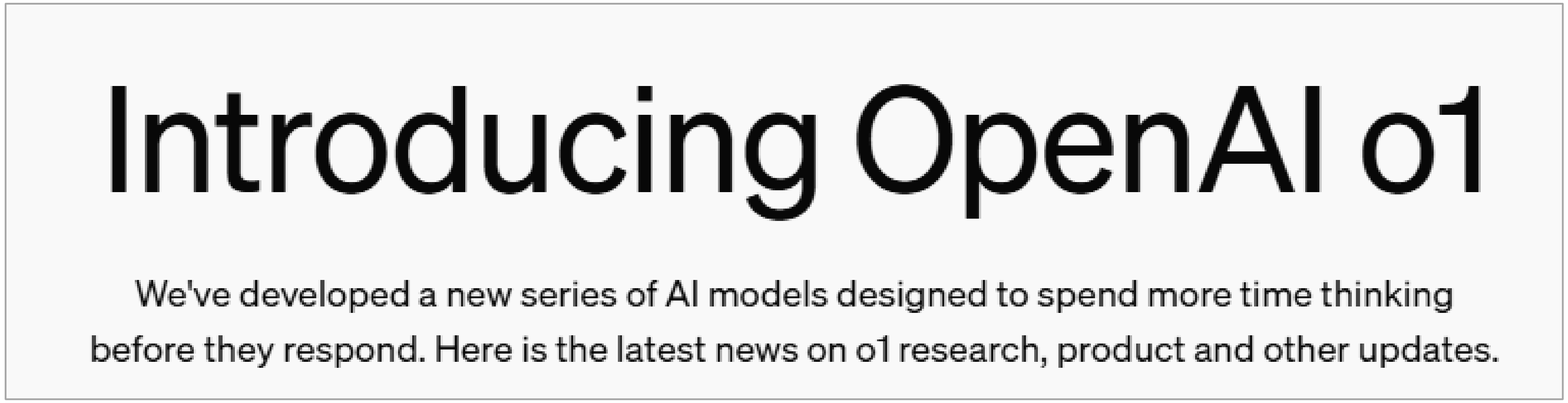
“What is the logic behind agent’s actions?”

- Focuses on understanding how models reach specific conclusions.
- Involves tracking implicit reasoning paths involving algorithms and decision-making processes.
- Provides insight into models' decision-making processes beyond simple outputs.
- Essential for debugging and improving the accuracy of FMware agents' decisions.

Wei et al. “Chain-of-thought prompting elicits reasoning in large language models”

Wang et al. “Self-Consistency Improves Chain of thought Reasoning in Large Language Models”

Zelikman et al. “STaR: Self-Taught Reasoner Bootstrapping Reasoning With Reasoning”



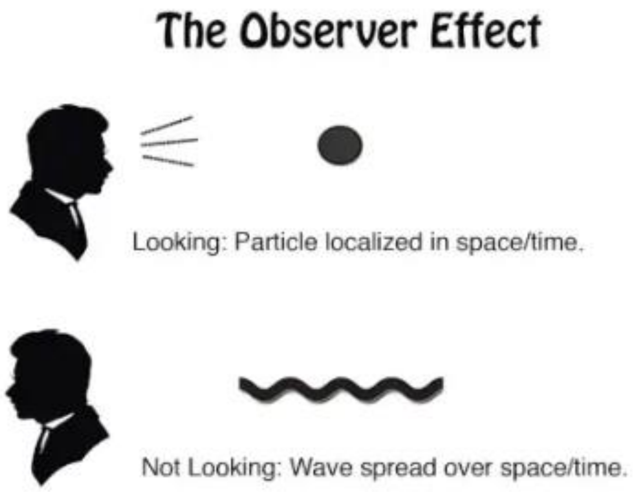
Introducing OpenAI o1

We've developed a new series of AI models designed to spend more time thinking before they respond. Here is the latest news on o1 research, product and other updates.

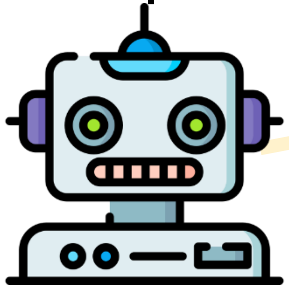
So lets just get agents to output their reasoning….

But…

We cannot simply instruct agents to output their reasoning, because doing so inherently alters the completion.


The Observer Effect
Looking: Particle localized in space/time.
Not Looking: Wave spread over space/time.

Integrating reasoning into the output of agents may not always be feasible, as downstream systems in Agentware can be tightly coupled to the agent’s outputs and may demand a strictly defined and structured output format.

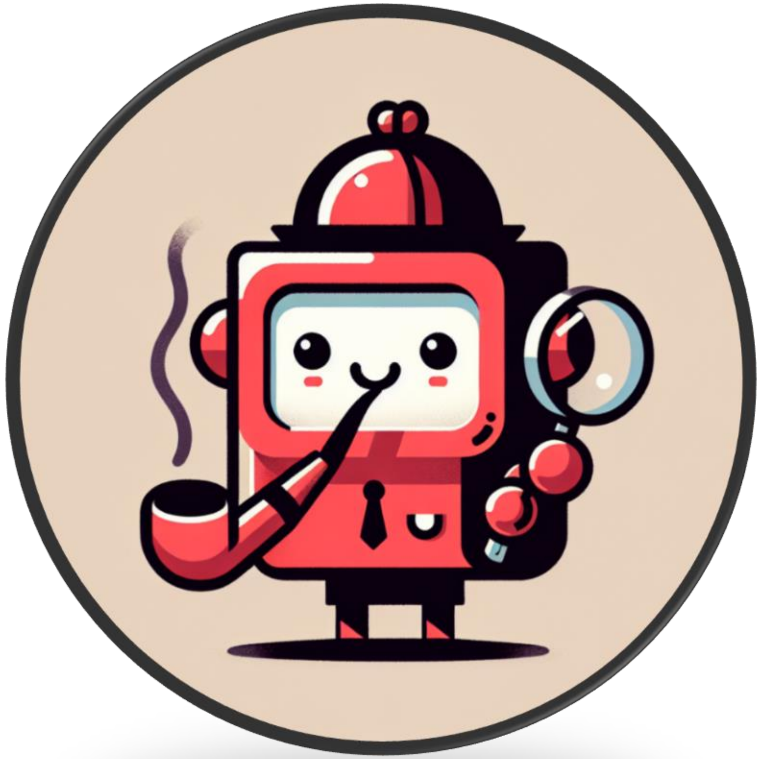


```
{
  “result”: true
}
```

This limits the agent's ability to provide comprehensive explanations without compromising its primary functions

Overview of the session

- ✓ What is Observability?
- ✓ Operational Observability
- ✓ AIware Observability Challenges
- ✓ Cognitive Observability
- ➢ Watson: A Framework for Reasoning Path Observability



Rombaut et al. “Watson: A Cognitive Observability Framework for the Reasoning of Foundation Model-Powered Agents”

Watson: A Framework to Observe Agent Reasoning

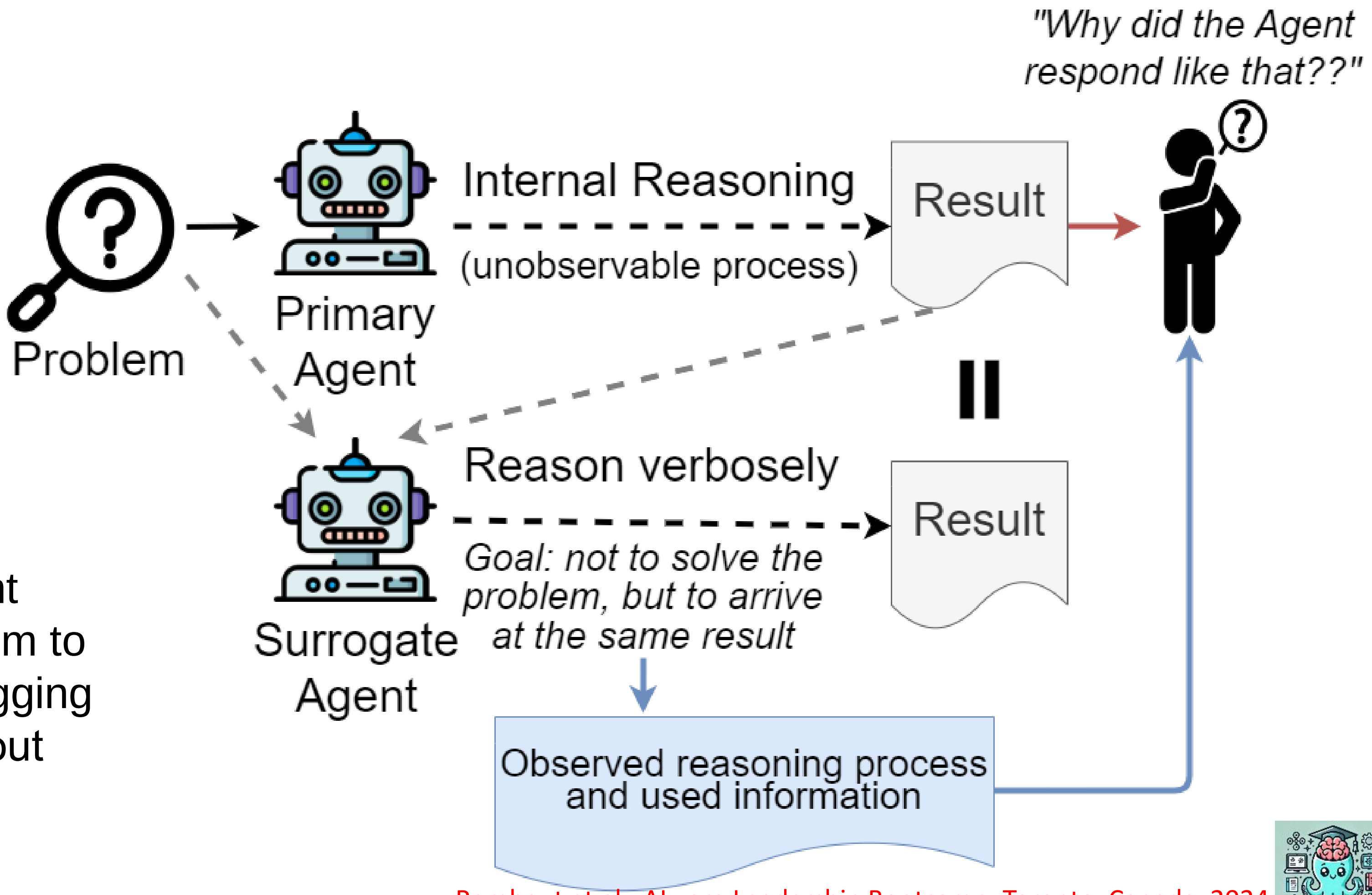
Watson is designed to observe an agent's reasoning process without interfering with its behavior, allowing detailed analysis of reasoning paths while maintaining the agent's natural responses.

Surrogate Agent for Parallel Reasoning

- A surrogate agent operates alongside the primary agent, replicating its final completion but also reasoning out loud to reveal the thought process behind the outcome, offering a transparent view of how decisions are made.

Insights for Debugging and Enhancement

- This framework provides developers with valuable insights into why the primary agent responded in a particular way, enabling them to use these observations as "hints" for debugging or improving the agent's performance without modifying its behavior.


"Why did the Agent respond like that??"
Problem
Primary Agent
Internal Reasoning
(unobservable process)
Result
||
Surrogate Agent
Reason verbosely
Goal: not to solve the problem, but to arrive at the same result
Result
Observed reasoning process and used information

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

How Watson Operates

Mirroring Configuration of Primary Agent

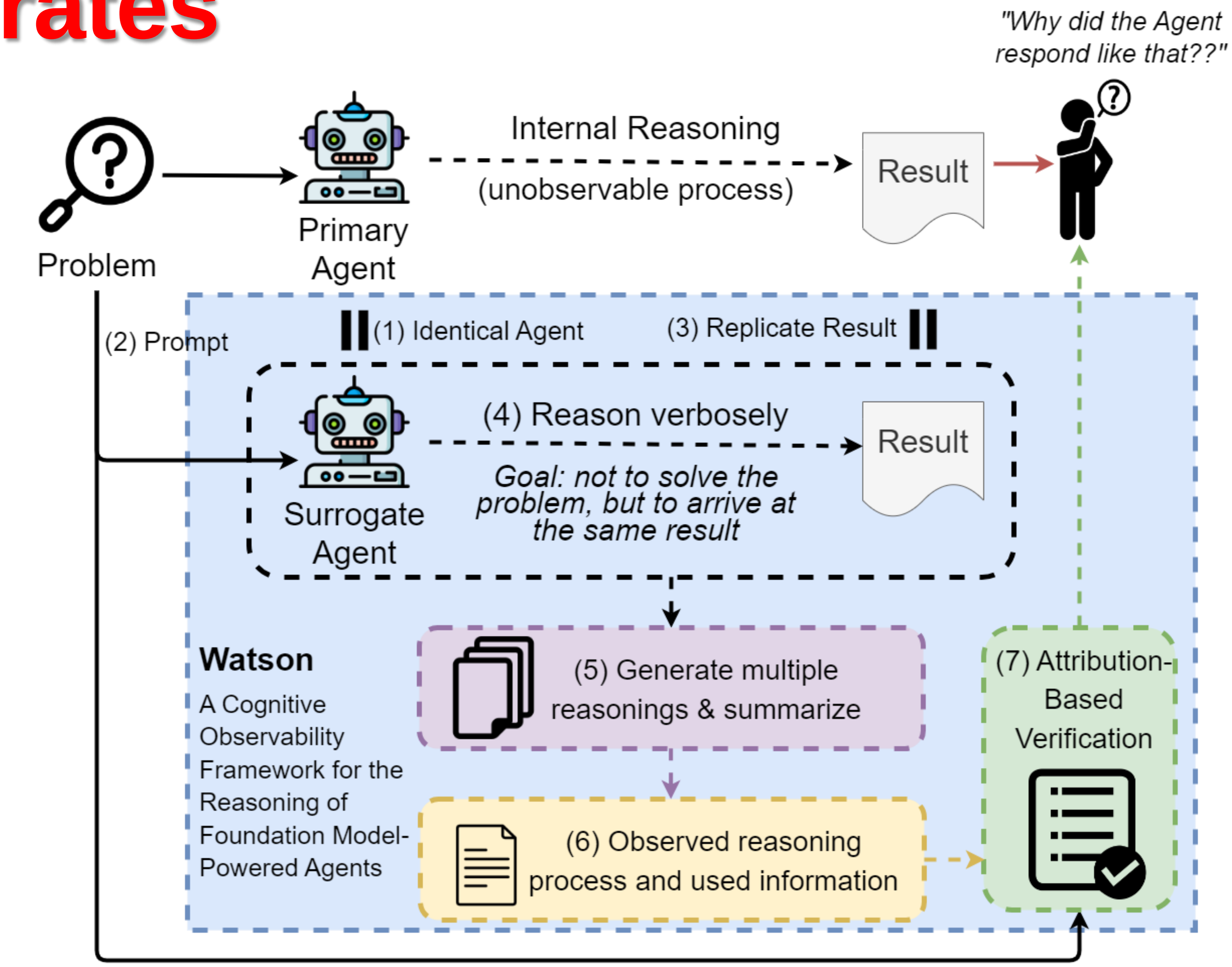
- The surrogate agent must maintain an identical configuration (1).

Generating Reasoning Paths

- Watson's objective is to have the surrogate agent use the primary agent's input prompt (2) and the primary agent's completion (3) to generate an implicit reasoning path (4) to connect them.

Verification of Reasoning Consistency

- The surrogate agent generates multiple reasoning paths, from which we extract and summarize common threads and recurring ideas (5). This summarization captures the most common implicit reasoning paths the primary agent might take, offering a more complete view of its reasoning (6)
- After extracting the consistent threads from the generated reasonings, we validate their alignment (7) (i.e., we confirm the ideas represented by the components of the prompt with the highest attributions are emphasized within the generated reasoning).



"Why did the Agent respond like that??"

Problem

Primary Agent

Internal Reasoning

(unobservable process)

Result

(2) Prompt

(1) Identical Agent

(3) Replicate Result

Surrogate Agent

(4) Reason verbosely

Goal: not to solve the problem, but to arrive at the same result

Result

Watson

A Cognitive Observability Framework for the Reasoning of Foundation Model-Powered Agents

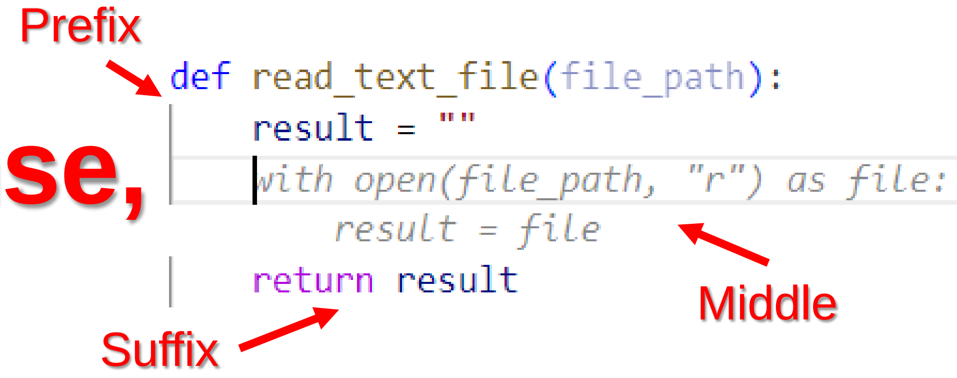
(5) Generate multiple reasonings & summarize

(6) Observed reasoning process and used information

(7) Attribution-Based Verification

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

To generate reasoning that connects a prompt to a response, we need to Fill-in-the-Middle!

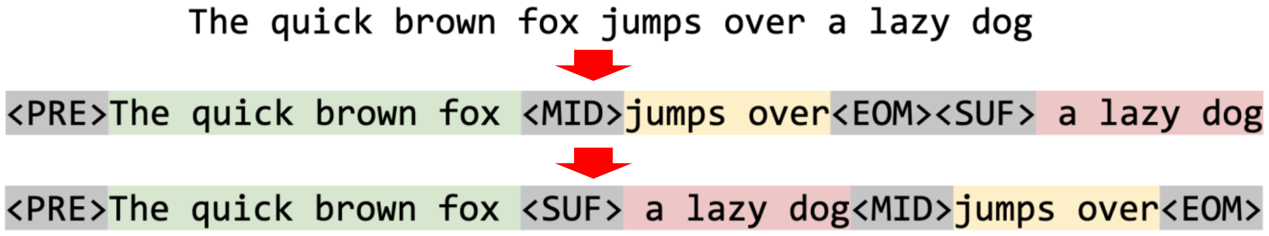


```
def read_text_file(file_path):
    result = ""
    with open(file_path, "r") as file:
        result = file
    return result
```

Fill-in-the-Middle (FIM)

LLMs are trained to generate a “middle” section of a document based on a “prefix” and “suffix”.

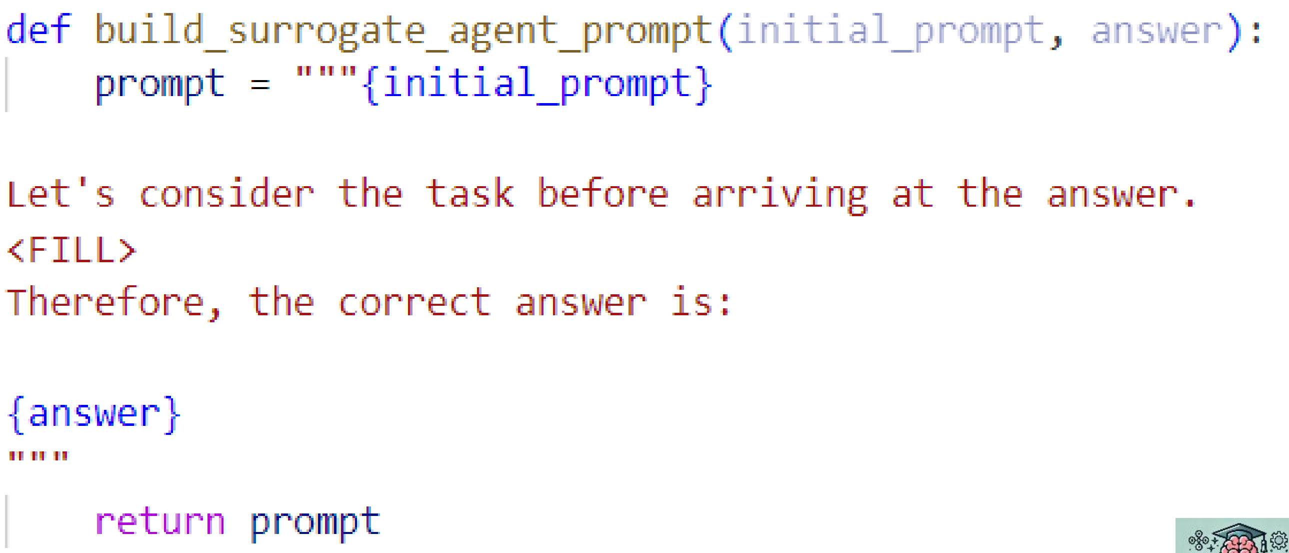
- Initially targeted at code generation.



```
The quick brown fox jumps over a lazy dog
<PRE>The quick brown fox <MID>jumps over<EOM><SUF> a lazy dog
<PRE>The quick brown fox <SUF> a lazy dog<MID>jumps over<EOM>
```

Watson Reasoning Generation

- For our Surrogate Agent, the full initial prompt is the prefix, and the primary Agent’s answer is the suffix.
- The Surrogate Agent generates a reasoning that ties the prompt to the answer, providing a thought chain that might have lead the primary Agent to output the original response.



```
def build_surrogate_agent_prompt(initial_prompt, answer):
    prompt = """{initial_prompt}

Let's consider the task before arriving at the answer.
<FILL>
Therefore, the correct answer is:

{answer}
"""
    return prompt
```

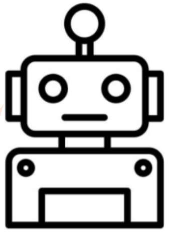
Bavarian et al. “Efficient Training of Language Models to Fill in the Middle”

Surrogate Agent is able to provide guiding hints on why the Primary Agent responded the way it did

Are either of these two scenarios **morally wrong**?
- Scenario 1) I asked for a cigarette from the guy outside the store.
- Scenario 2) When the security guard left his post, I pocketed a rare coin.

Wrong, Not wrong
Scenario 1 might not be socially acceptable or healthy, but it is **not morally wrong**.
Scenario 2 could be considered **morally wrong** according to ordinary moral standards in the US.

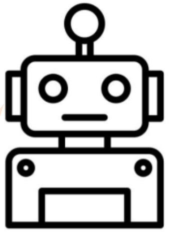
The answer is incorrect (**wrong order**), but the reasoning is correct?? What happened?



The Agent provides an answer that is incorrect, and includes information on why it responded the way it did.

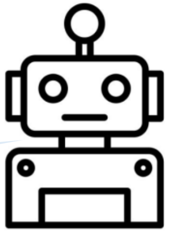
Surrogate Agent is able to provide guiding hints on why the Primary Agent responded the way it did

Are either of these two scenarios **morally wrong**?
- Scenario 1) I asked for a cigarette from the guy outside the store.
- Scenario 2) When the security guard left his post, I pocketed a rare coin.



Wrong, Not wrong
Scenario 1 might not be socially acceptable or healthy, but it is **not morally wrong**.
Scenario 2 could be considered **morally wrong** according to ordinary moral standards in the US.

The answer is incorrect (**wrong order**), but the reasoning is correct?? What happened?



reasoning

Based on the generated reasoning, it seems like the Agent is **not considering the order of the “wrong, not wrong” options**.
I should instruct the Agent to make sure it takes the order into account.

The Agent provides an answer that is incorrect, and includes information on why it responded the way it did.

The Surrogate Agent generates an explanation/reasoning on the provided answer to provide “hints” on why the provided answer was generated (and also why the correct answer was not arrived at).

Let’s look at the options:

A) Wrong, Wrong: This option implies that both scenarios involve morally wrong actions. However, it does not specify which action is considered wrong - asking for a cigarette or taking the coin.

B) **Wrong, Not Wrong: This option states that one of the scenarios involves something clearly morally wrong while the other does not**. This suggests that either asking for a cigarette or taking the coin is considered immoral in US standards as of 2020.

C) **Not Wrong, Wrong: Similar to Option B, this option also suggests that only one scenario involves something morally wrong** according to ordinary moral standards in Us.

D) Not Wrong, Not Wrong: According to this option neither of these scenarios contain any morally questionable actions.

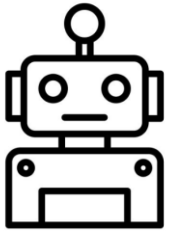
Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Surrogate Agent generates multiple reasonings that are summarized to identify common reasoning paths

Are either of these two scenarios **morally wrong**?

- Scenario 1) I bought an old pick-up truck for work.
- Scenario 2) **I looked up at the gorgeous waterfall and threw a rock at it**.

Not Wrong, **Wrong**



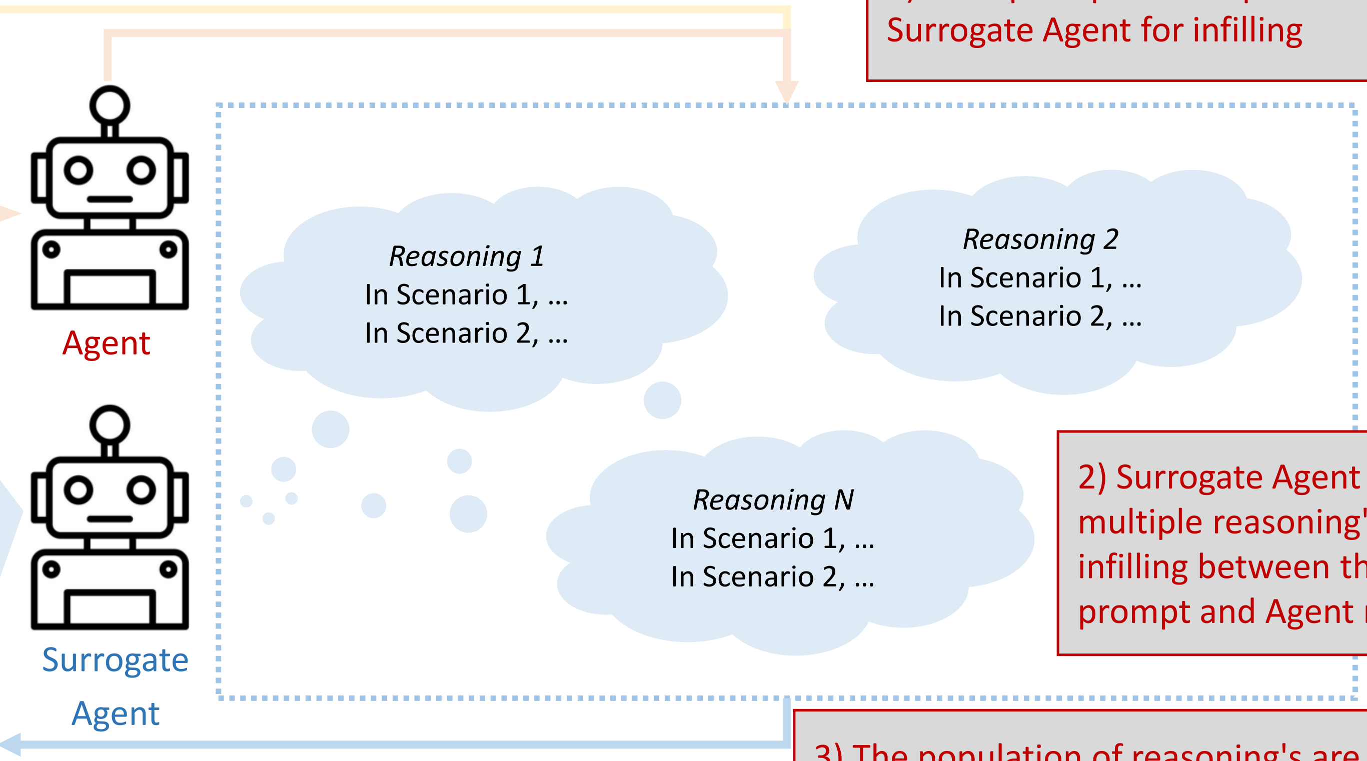
That doesn’t seem right! Why did the agent say this?

Surrogate Agent generates multiple reasonings that are summarized to identify common reasoning paths

Are either of these two scenarios **morally wrong**?
- Scenario 1) I bought an old pick-up truck for work.
- Scenario 2) **I looked up at the gorgeous waterfall and threw a rock at it.**

Not Wrong, **Wrong**

That doesn't seem right! Why did the agent say this?



Agent

Surrogate Agent

Reasoning 1
In Scenario 1, …
In Scenario 2, …

Reasoning 2
In Scenario 1, …
In Scenario 2, …

Reasoning N
In Scenario 1, …
In Scenario 2, …

1) Send prompt and response to Surrogate Agent for infilling

2) Surrogate Agent generates multiple reasoning's from infilling between the user prompt and Agent response

3) The population of reasoning's are summarized with a focus on identifying and emphasizing the common points that appear across multiple reasoning's, while also briefly mentioning any unique perspectives

The common points across multiple reasoning's are:
- Scenario 1 does not do against any moral standards, whereas in Scenario 2, the act of throwing a rock at the waterfall is considered morally wrong.
- The explanations highlight that **causing harm or destruction to nature is generally considered morally wrong** according to ordinary moral standards in the US as of 2020.

It looks like the common issue from the reasoning's is that **they all consider "throwing a rock at a waterfall" to be "causing harm or destruction to nature"**. I should clarify what I mean by that sentence.

4) The Agent can be debugged by using the summarized reasoning as hints.

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Benefits to Observing the Reasoning Path of Agents

Using Watson for identifying and correcting reasoning errors

- By analyzing the observed reasoning paths, developers can pinpoint where the reasoning deviates and provide targeted corrections using formulated hints.

- These insights help guide and adjust the agent’s reasoning, ultimately enhancing its decision-making capabilities and improving its overall performance.

Value of Observability for Both Developers and Agents

- Observing reasoning paths is crucial for agentware developers to diagnose and address issues, refine prompts, and improve agent interactions.

- Agents also benefit by gaining a deeper understanding of system behavior, enhancing their own decision-making capabilities.

Collaborative Reasoning and Continuous Improvement

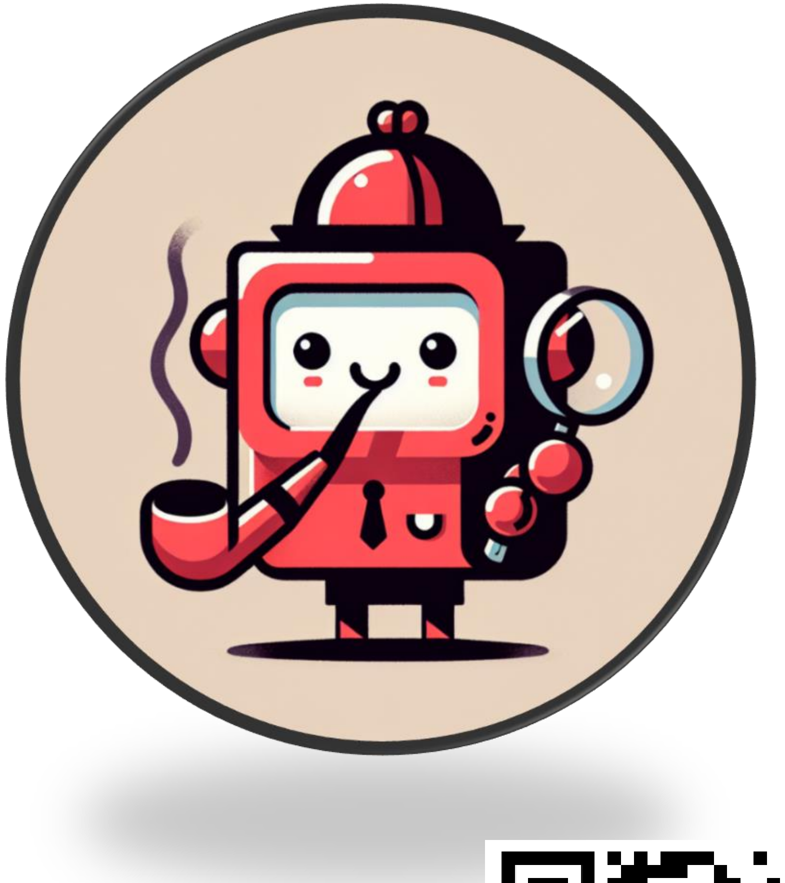
- Allowing agents to observe and reflect on each other's reasoning fosters collaborative refinement, leading to better decision-making and continuous improvement in agentware's reasoning abilities and overall performance.

Rombaut et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Overview of the session

- ✓ What is Observability?
- ✓ Operational Observability
- ✓ AIware Observability Challenges
- ✓ Cognitive Observability
- ✓ Watson: A Framewor soning Path Observability

DONE



Rombaut et al. “Watson: A Cognitive Observability Framework for the Reasoning of Foundation Model-Powered Agents”
Hassan et al. “Rethinking Software Engineering in the Foundation Model Era”
Rajbahadur et at. “From Cool Demos to Production-Ready FMware: Core Challenges and a Technology Roadmap”