The Ticking Timebomb of Observability Expectations

...and how to defuse it

David Caudill
Sr Lead Software Engineer, SRE
Capital One Enterprise Reliability

Disclaimer:

This talk is comprised of my own personal opinions and research, and does **not** represent the opinions or positions of my employer.

Agenda

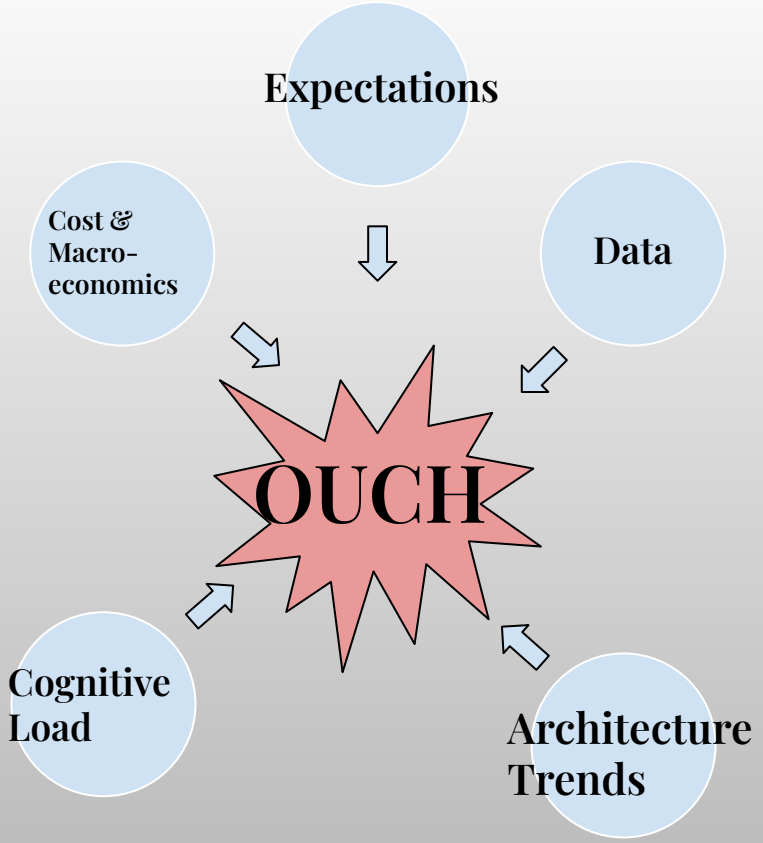
1. About Me
2. The Problem
3. Tempting, Bad Ideas
4. Ideas and Solutions
5. Conclusion

About Me

- Professionally trained as a teacher, ophthalmic surgical assistant, driver, etc.
- Been in tech about 15 years: IT, Software Development, QA, SRE and lots of management.
- Experience with finance, biotech, education, healthcare, adtech, and tons of SaaS.
- Things I love: pugs, folk and tribal art, electronic music, and cybernetics.
- I work in NYC.

The Problems With Observability:

- The Loudest Voices are Vendors
- Unrealistic Expectations
- Lousy Data & Too much of it
- Complex Architectures
- High Prices/Costs
- High Cognitive Load



Expectations:

- “Monitor Everything” and sort it out later: Store billions of logs, traces, and metrics on the off chance we might need them.
- Retrieve them almost instantly and error-free
- Use this data to solve real world problems with no training or documentation
- Have this data available even in an outage for a cloud provider
- Do all of this for less than it actually costs

The Data:

You likely have *millions of line graphs,* and *billions* of logs.

This data is **mostly junk**, it's highly contextual…and it's all out of context.

You probably have no utilization information on this data, so you have no idea what's being used by whom.

Some of the data is extremely important and useful.

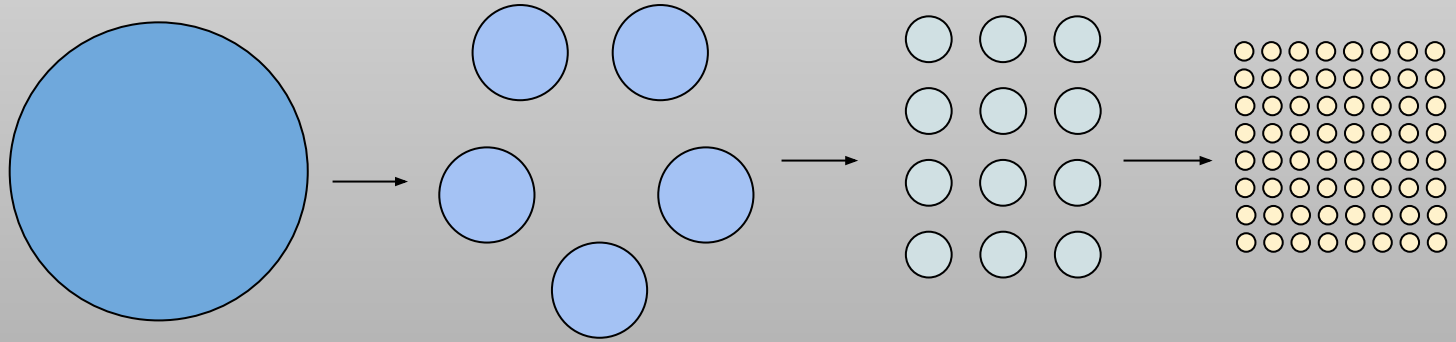
It's often very difficult to browse or play in this data.

Architectures:

Architectural trends from Service Oriented architectures into Microservices and Serverless have introduced *massive* IPC overhead.

What was once a method call could now be a call to a separate service, or function, on any one of a dozen platforms.

Every call is potentially instrumented to forensic detail, and in some cases, even instrument startup/tear down actions.



The Price: 💰

The end of the 2010's cheap money has stopped the flow of VC investments that were subsidizing prices for products.

Now vendors need to…make money.

Something has to give.

Source: Board of Governors of the Federal Reserve System (US)

Release: H.15 Selected Interest Rates

The Price (cont'd):

Storing all this data is not cost effective **even for the vendors themselves**, who have no magic for avoiding massive cloud costs.

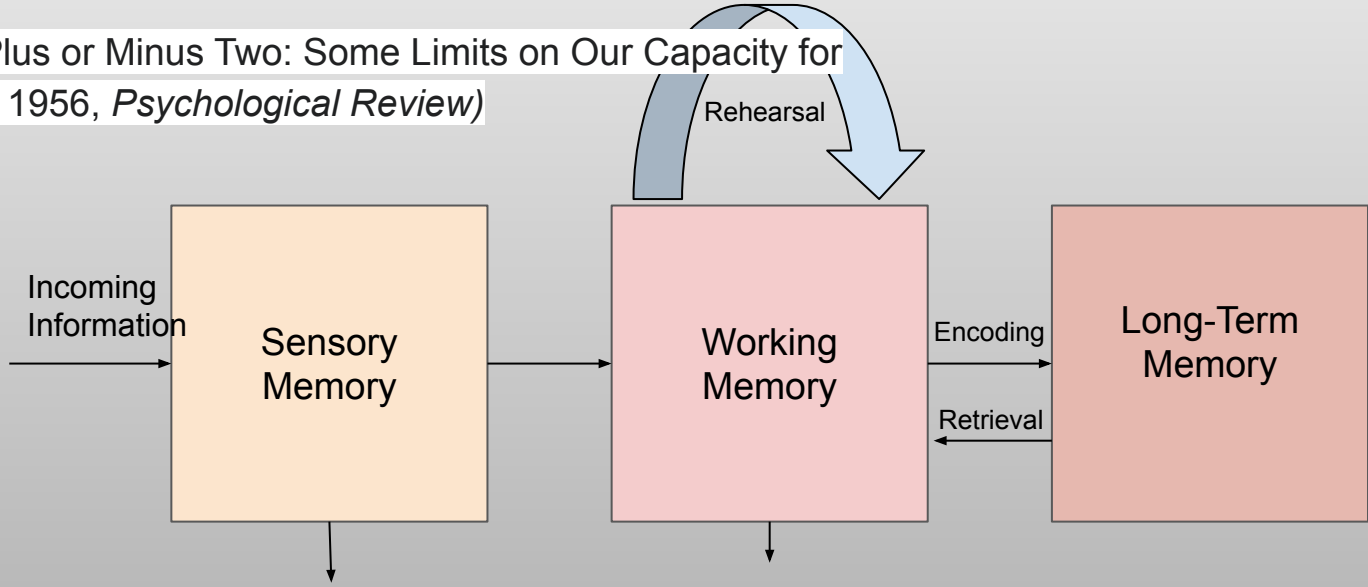
Sooner or later, somebody from your finance team is going to ask a very good question: How can these tools *possibly* be worth what you're paying for them?

Delusional: Can’t Ignore Cognitive Load Theory 🙈

Cognitive Load/Mental Workload is “how much stuff are you holding in your head at once”.

Miller’s Law:

"The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information" (Miller, 1956, *Psychological Review)*



Cognitive Overload 🧠🔥

Unsurprisingly, your capacity goes **down** because of outside noise and stressors.

Like… a late night incident, executives on incident calls, a pile of poorly named dashboards, 200 people on an incident bridge, and millions of metrics.

Are your dashboards taking this into consideration?

NOPE. 🙅

Dead Ends and Well-Worn Paths to Nowhere

- You already have more data than you know what to do with.
- The same vendors and tools that got you into this mess are going to sell you “solutions” for getting out of it.
- Vendors will appeal to your ego and convince you that you can process more information than humanly possible.
- AI/ML solutions seem designed to expend maximum carbon for generating spurious *correlations.*
- “AI” is not going to save you from this, and it’s not going to stay cheap, either.

Nobody/nothing can understand your system *for you.*

...What now?

Given external shifts and focus on working with greater efficiencies, it's safe to say that *most* of us in SRE will have to reduce our spend and utilization on these platforms:

- Without losing what's important.
- Without getting worse outcomes in incidents.
- With only limited advice from vendors because of the obvious conflict of interest.

The only solution here is to "do the work" of making sure people actually understand this data, and can operationalize it.

The Hard Work of Constructing Meaning

What is MEANING in this context?

- Your team understands the consequences of a given metric or alert
- Your team changes behavior based upon what they see.
- Your team can put the data into context.
- Your team uses the data to identify problems, and to verify that they've been solved.

Can you *really* expect anybody to engage with this data while it's meaningless to them?

Is it even possible for this data to be valuable if you cannot do these things?

Why is meaning “constructed”?
Why can’t I just tell people what something means?

You have to do **both!** Your engineers need to interact with this data, and with somebody who understands it well.

They need to *play* in the data in order to “make sense of it.” How does it fit together with what they already know? How is it structured?

In other words, you have to *tell them* and they have to actively *learn it.*

Our work here is to make that learning as easy as possible.

Work Backward from a Vision

Think like a product manager - identify a few scenarios these tools should cover for you. Thinks about the personas involved in each.

- Incident Response
- Performance Degradation
- High level Status Communication

Fighting Cognitive Overload

Quick Tips:

- DO design dashboards for a new developer with a migraine.
- DO take advantage of note widgets in dashboards.
- DO ensure similar items are grouped logically and use a visual container if your tools provide it.
- DO decide on a few very important dashboards you will share with your team.
- DO put someone in charge of making the data meaningful.

- DON’T pack status dashboards with more info than you need.
- DON’T build a new dashboard every single time you want to know something.
- DON’T build dashboards in response to every single incident.
- DON’T break Conway’s Law

Separate Status from Diagnostic Information

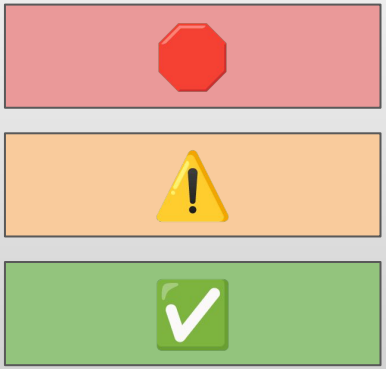
Status Information: CHECK ENGINE

How are things going right now for my customers?

IS something wrong **right now?**

Low Cognitive Load

(Status information is what good SLOs are made of!)



Diagnostic Information: OBD2 Code

What's happening in my system? WHY is something going wrong? When did it go wrong?

High Cognitive Load

Start with what is important, not what's easy

- "Turnkey" integrations and agents have a way of filling these systems with metrics and giving you the illusion that work has been done.
- Understand what an application *does* and instrument it until you know it's done that well.
- Don't confuse rolling out a new tool with constructing meaning.

Examples:

"When a customer clicks the Check Out button, what is the error rate? How long does that take?"

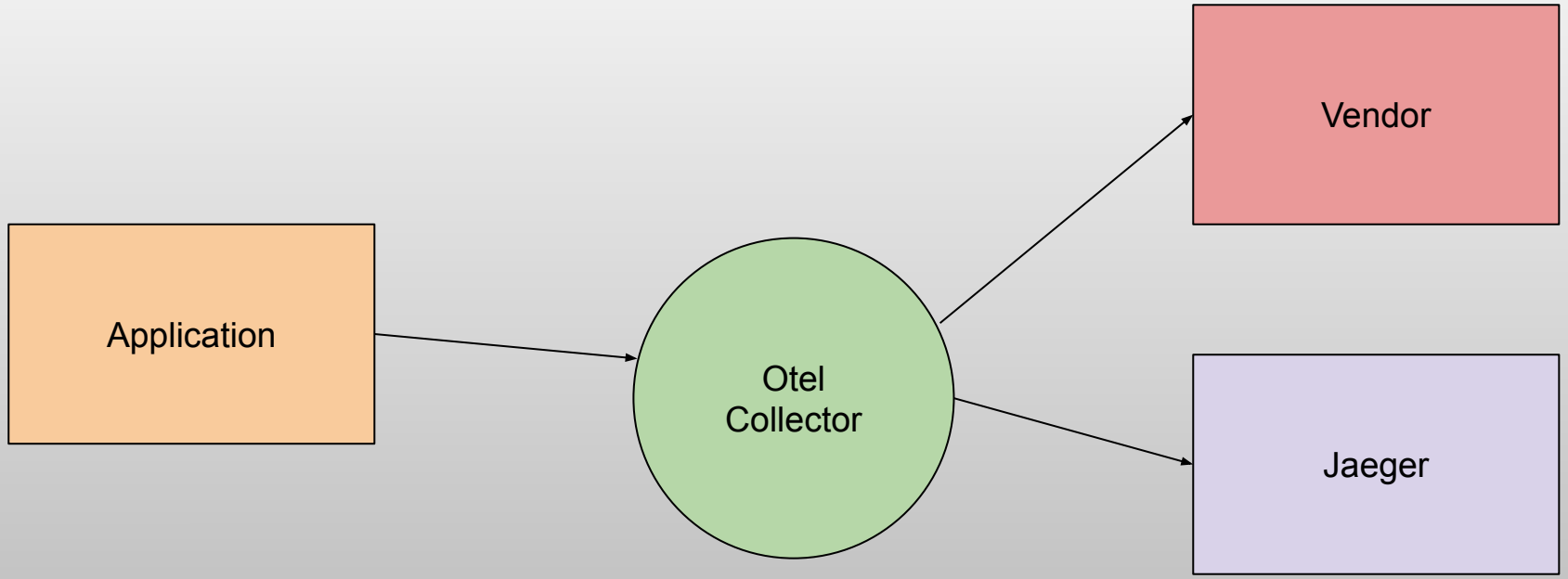
"When we get an error at this endpoint, what actually happens to the customer?"

Managing Costs

Managing Costs: Know your Vendor

- Billing schemes are cryptic and complex.
  - There is no free lunch here - if you find a loophole, it's waiting to be closed. You'll **have** to reduce what you send.
  - The best way out is to understand where value vs cost aren't making sense for you, and reduce there
  - Special billing deals have a way of becoming regular billing overnight, which can come with surprises.
- Spend time with the bill and understand where money is going.
  - This takes engineering hours to investigate, it's worth it.
  - If it's not detailed, **consider another vendor.**
  - **Send less. Retain less.**

The Role of an OTel Collector:



Question Your Assumptions:

Must you have this problem?
Is this *your* problem, or is it just an *interesting* problem?

- Do we really have **web scale**?
- Do we really **use** this?
- What problems does data this solve?
- If we did not know this, how would my actions be different?
- Could I still use this information with slower access? Sampling?

Most importantly: Am I blind to these assumptions because they serve my ego?

Conclusion:
How will you know if things are getting better?

Ask Questions!

Examples:

“Hey I noticed we have an alert set on CPU. What does it mean for customers when that alert has fired?”

“These two things are next to each other on your team’s important dashboard. How do they relate to each other?”

“What’s the most important thing for me to monitor about this application?”

Ask several people, compare their answers.

Evaluate Behaviors

Review post mortems and look for cases where:

- The data was used to make a decision.
- The data was used to validate a decision.

Anti-patterns 🚩:

- “I tried this myself and it appears to be working”
- Incidents declared because of an executive’s experience

Review The Observability Bill

- Does it continue to stumble higher? Why?
- Are you still having “accidents” where your bill goes ballistic?
- Are the value to your team and the cost getting closer to aligned?
- Do you understand it?
- Do the people doing expensive things understand them?

Thanks!