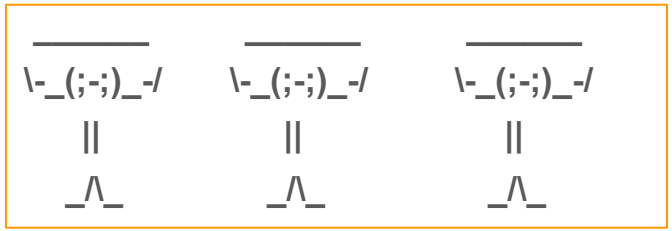
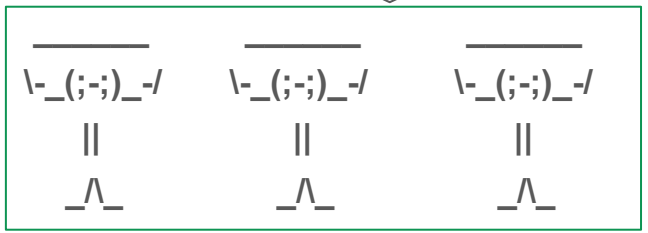
The best SREs seem to be the ones without the title -- and what we can do about it?

Kishore Jalleda - **unStruct.ai**

We write more code; we are the better SREs!



What a joke. We run production; we are the best!



To make it clear: this talk is not about who is better :-)

Let's start with a real-world example

A ??? in the east coast causes a global SEV1

- An issue, which recovered in minutes, caused a cascading failure and partial outage of a distributed app.
- Failure symptoms pointed to the wrong teams causing multiple reassessments and delays.
- Insufficient alerting and platform observability made it hard to pinpoint the root of the problem.

Insufficient_Platform_Observability

Multiple_Reassigments

Total teams involved in incident response.

7

SRE Team

Eng Team 2

Eng Team 1

Other Eng Team 1

Other Eng Team 2

Triage Team

Internal Customer Team

Basic Incident Metadata

TTR: <1 hour.

Participants: 20+

Toil~: (20 x 2 TTR + 20 x1 PIR + 10 X 2 PIT) ~= 80 hours.

Advanced Metadata

No_of_reassignments: 4

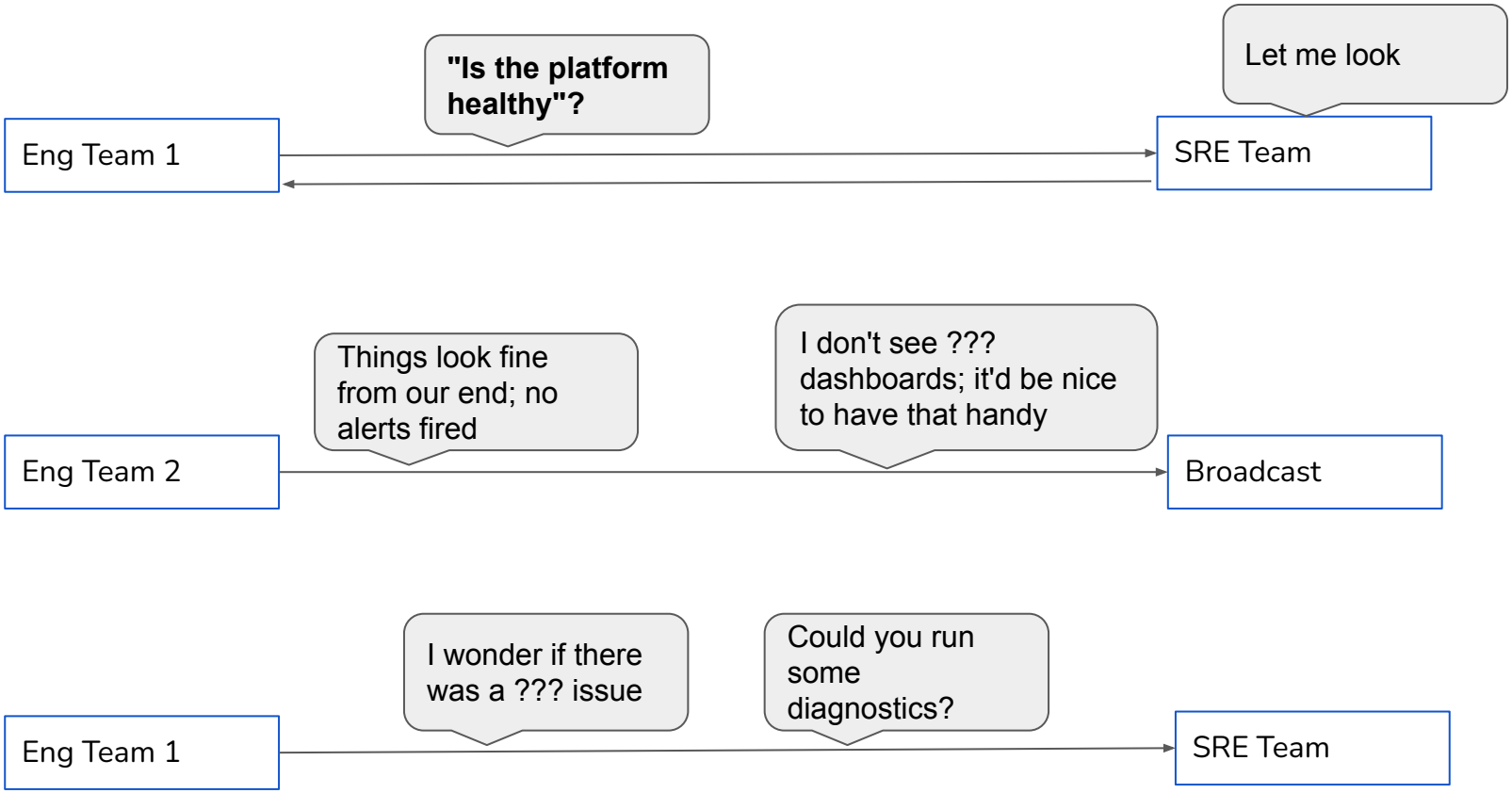
Amount_of_misalignments: 2

No_of_adhoc_debuggings: 5

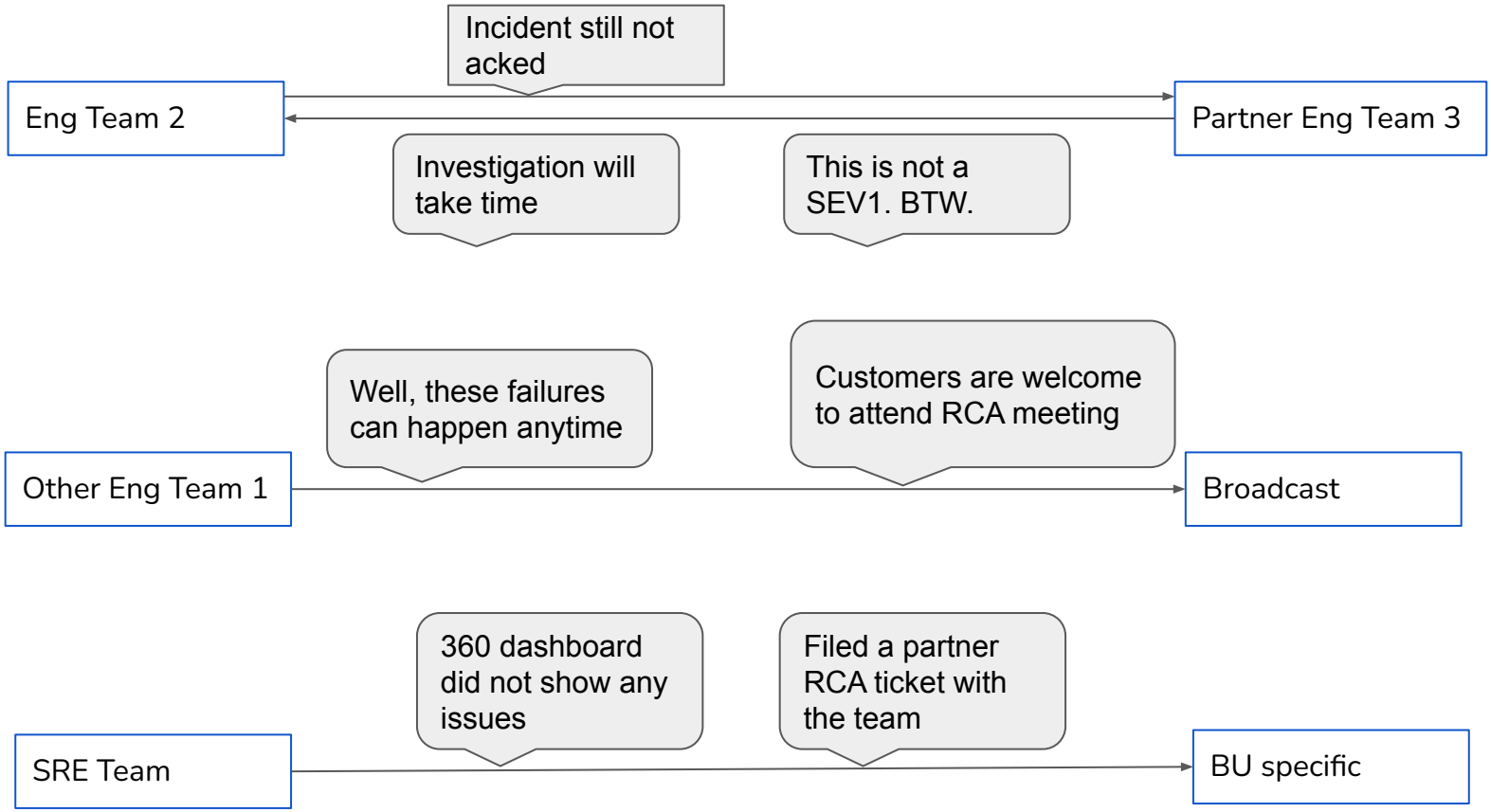
No_of_frustrated_people: 20

No_of_tools_that_did_not_work_during_mitigation: 3

R/T Conversations during the incident



R/T Conversations during the incident



Any guesses on what the issue was?

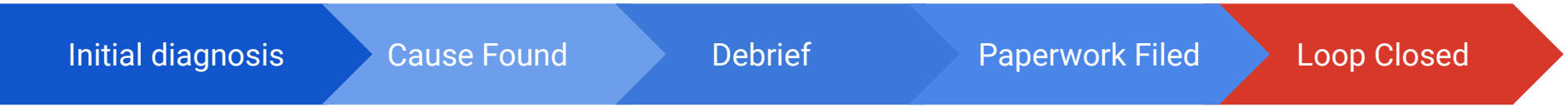
Any guesses on what the issue was?

It's the Network.

The issue was a **fiber cut** in a data center.

Here's how this could have gone better (Hypothetically)

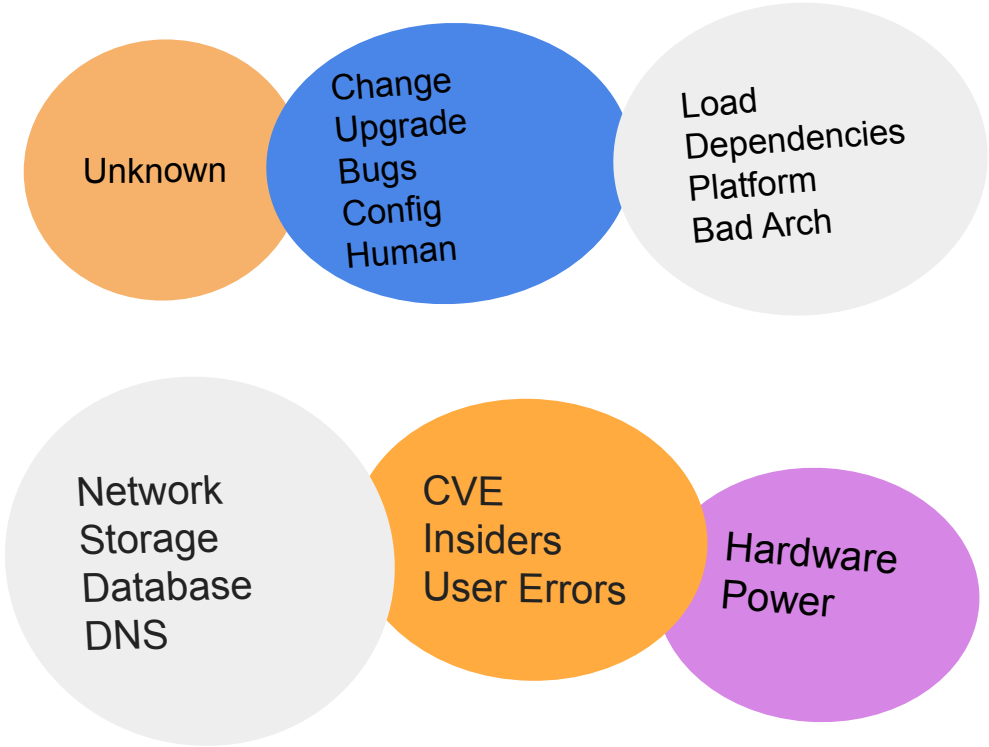
Hypothetical Result: Total time spent less than 30 minutes, with a handful of folks. Not 20+.



"is the platform healthy?"	Bad link from the fiber cut. Incident mitigated after an app tier restart	Add resilience to service-to-service calls	File a partner RCA ticket with network.	Cadence exists for managing risks. Loop closed in a few weeks.

The_Problem:

Why does the cloud stop computing?



Analysis of 1247 headline news and public postmortem reports, that detail 597 unplanned outages that occurred within a 7-year span.
https://ucare.cs.uchicago.edu/pdf/socc16-cos.pdf

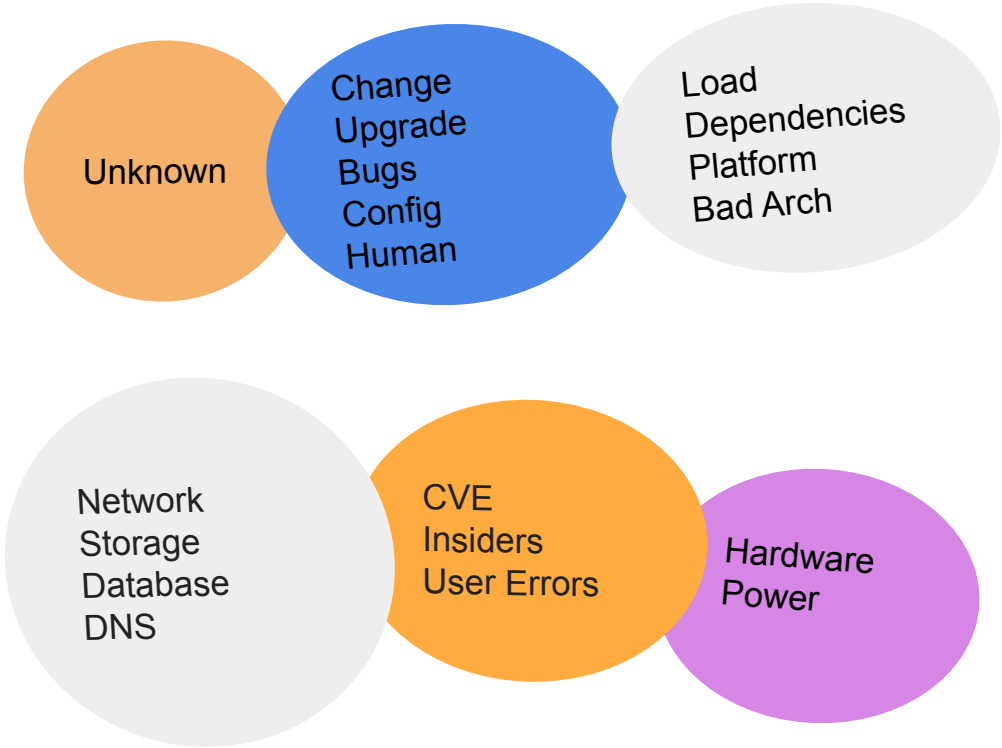
Y = ?
Any guesses?

Only Y% of incidents can be mitigated immediately

based on a study at Microsoft:
https://www.cse.cuhk.edu.hk/lyu/_media/conference/zchen_esecfse2020_towards.pdf

The_Problem:

Why does the cloud stop computing?



Analysis of 1247 headline news and public postmortem reports, that detail 597 unplanned outages that occurred within a 7-year span.
https://ucare.cs.uchicago.edu/pdf/socc16-cos.pdf

Y = 90

Only 90% of incidents can be mitigated immediately

based on a study at Microsoft:
https://www.cse.cuhk.edu.hk/lyu/_media/conference/zchen_esecfse2020_towards.pdf

Three major patterns in those 10% of the incidents

Pattern #1:

Service/Resource dependency graph is inadequate

- Service_dependancy_map

Pattern #2:

Failure's symptoms insufficient to point to the responsible service team

- Who_to_call?
- What_alerts_to_fire?
- Multiple_assignments

Pattern #3:

Long time to identify possible causes and impact scope

- Platform_is_complex
- Too_long_to_detect

Who are almost always called for help?

Mary or Bob from that SRE-like team.

As a result of this complexity, Mary and Bob from that SRE-like team, or that central service are almost always called for help, if the failure symptoms are hard to pinpoint to a possible cause or a team.

Leading to burnout, frustration and productivity losses.

How did we get here?

Contributing Factors

Contributing_Factor_1:

Lack of understanding of what the true spirit of SRE is, leading to resource misallocation, inefficiencies and wasted efforts.

Contributing_Factor_2:

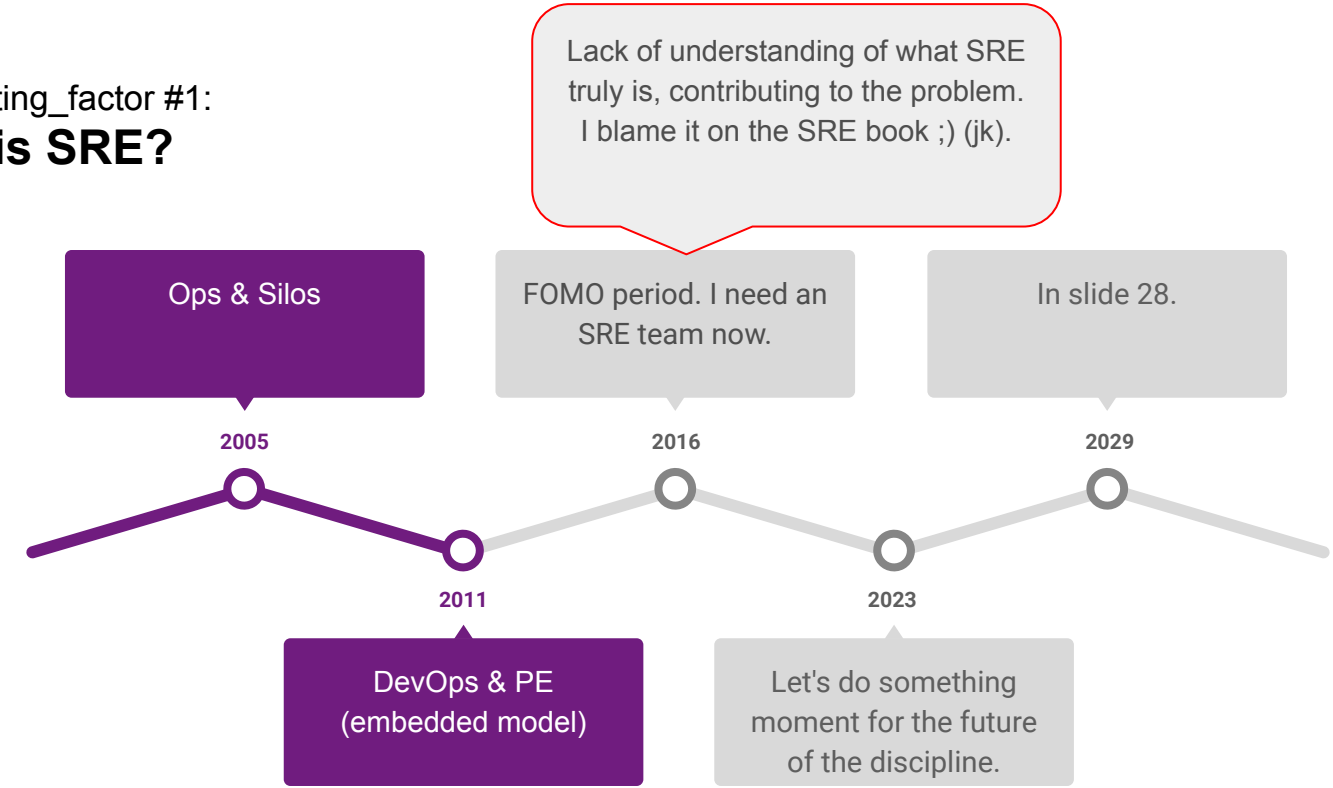
IC/Leadership misalignment; organizational politics leading to anti patterns and wrong incentives.

Contributing_Factor_3:

Tooling built by non practitioners, leading to low adoption, ROI and efficacy. External tools can do a better job to make things better; most seem to look and feel the same.

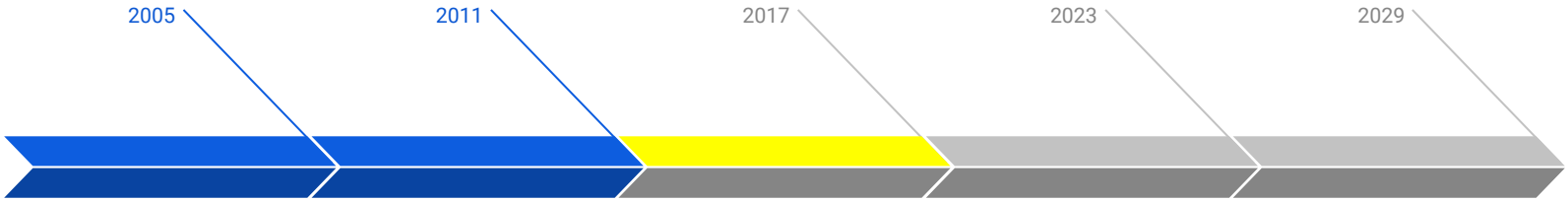
Contributing_factor #1:

What is SRE?



Contributing_factor #2 (What is SRE?):

Org Politics and IC/Leadership Misalignment



Mgmt 1.0

Tech figuring out management.

Good times; Lean, mean times

Good times. Tech started to boom. People worked hard.

Multiple layers & Hierarchies

I need 3 layers under me to manage 20 people. Or a VP/Dir having 2 directs.

(SRE) Management 2.0 Definition Time

Code will be the conduit between ICs and Managers.

This one is tricky. But.

The era of pure people manager roles might be over. Time to act now().

This period is likely when some bad decisions were made with hiring, vision and mission for SRE.

Contributing_factor #3:

Incomplete SRE Tooling

Just for fun exercise:

What if the SRE book was instead a…?

Any guesses?

Contributing_factor #3:

Incomplete SRE Tooling

Just for fun exercise:

What if the SRE book was instead a…?

Any guesses?

an…

SRE (SaaS) Platform

That would have been nice ;-)

Contributing_factor #3:

Incomplete SRE Tooling

Many were born during this period

On-call scheduling & Observability, ITIL, Bug Tracking, etc.

FOMO Moment

Enterprise SaaS bloat.

Crowded Market

Most seem to look and feel the same.

Most will likely be extinct.

Unless you focus on real problems and real solutions.

Contributing_factor #3:

Incomplete SRE Tooling

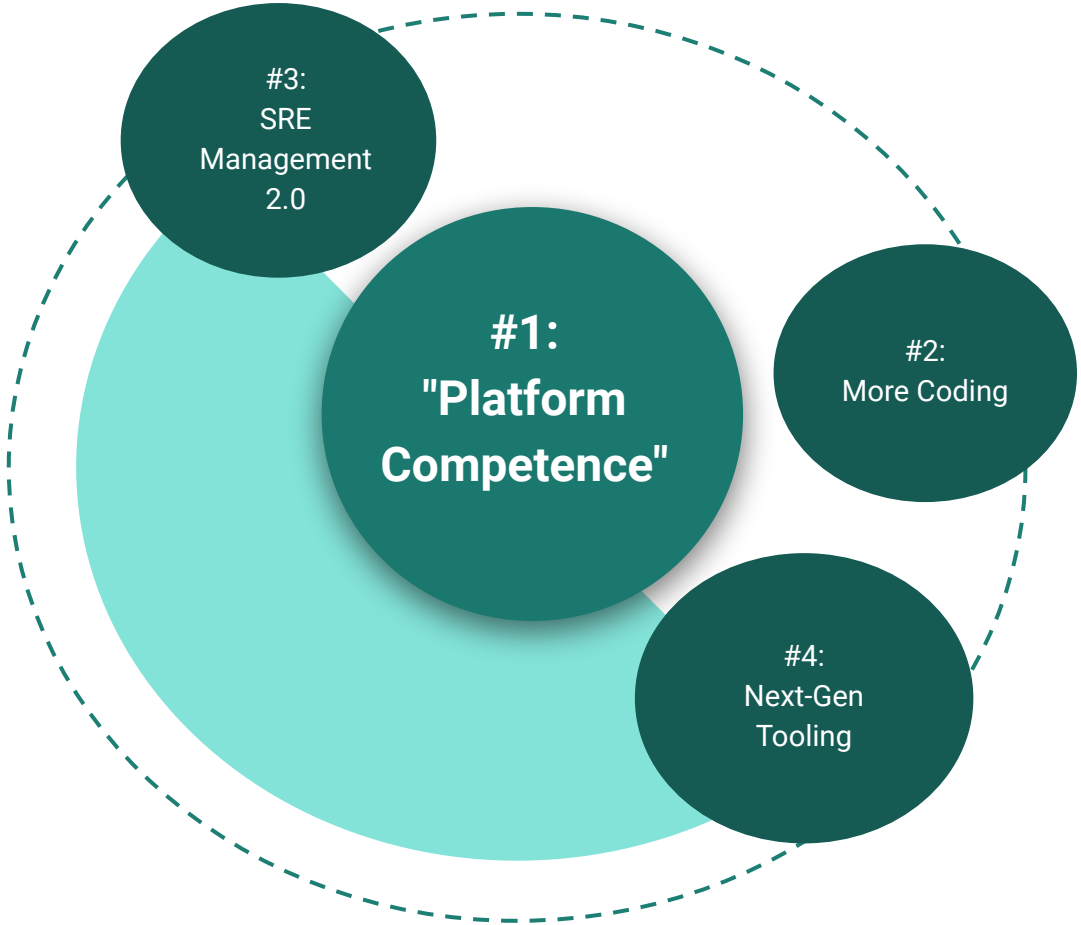
Most seem like they are working on those 90% use cases - But why?

- N+ variations of **Paging People**
- **Etc.**
- **Etc.**

{N} - left to your interpretation ;).

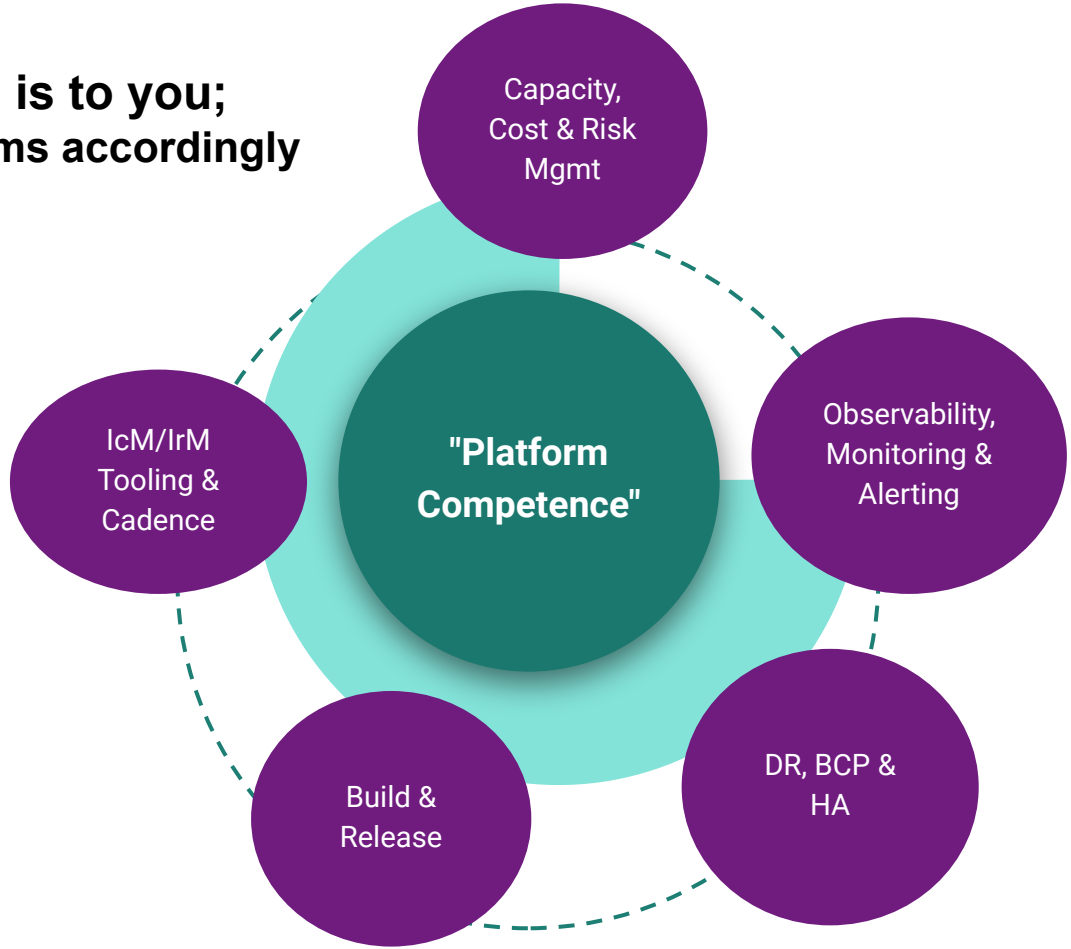
If you were to replicate a tool, at least add your secret sauce :).

Practical Solutions



Practical_solution #1:

Define what SRE is to you;
Structure your teams accordingly



Practical_solution #1:

"Platform Competence"

What is "Platform Competence"?

It is the ability to answer one simple question: "Is the platform healthy?"

Platform, in this context is the complex tree of dependencies that are required to build and operate your app/service, including but not limited to network, storage, databases, people, service owners, process, service/resource dependencies, middleware, abstractions, observability, monitoring, CI/CD, (L)LM Infra; and other components, that are needed to reliability, efficiently and securely operate an online service.

Note: Term platform competence not coined by me; it was cloned by someone (or a group of people) on my team at Microsoft.

Practical_solution #1:

"Platform Competence"

What is "Platform Competence"?

More importantly, it is something AI will take much, much longer to replicate, given the non-deterministic attribute of the nature of work it entails - which is having the ability to connect complex relationships, among the sea of abstractions and frameworks upon which modern software is built.

Making your platform more observable is a hard problem.

"*His awesome presentation on the future of programming states that in ~3 yrs, an AI robot engineer will cost < $10 per month and will do everything an engineer can today.*"

- Source: LinkedIn.

Note: Term not coined by me. It was by someone (or a group of people) on my team at Microsoft in 2018.

Practical_solution #2:

Coding Goals for Everyone

Everybody needs to code (to a varying degree, of course) -- **even leaders**. This is for developing empathy, stepping up and assisting the team when needed, for proper rep at promo time) -- with the macro goal of creating a true SRE culture where the practitioners can focus on platform competence. **Start with some official coding goals/KPIs.**

Need more management and leadership participation on the front lines -- and attending more post-incident reviews.

Practical_solution #2:

Toil Tracking and Reporting

Toil is evil

And should not be tolerated; it must be properly tracked and taken seriously by management, to free up time for coding and other things that matter.

Mindset shift

SREs must learn to **say no** and share the love; must be comfortable enforcing SLOs and error/alert/toil budgets. Again, so they can focus on things that matter.

Practical_solution #3:

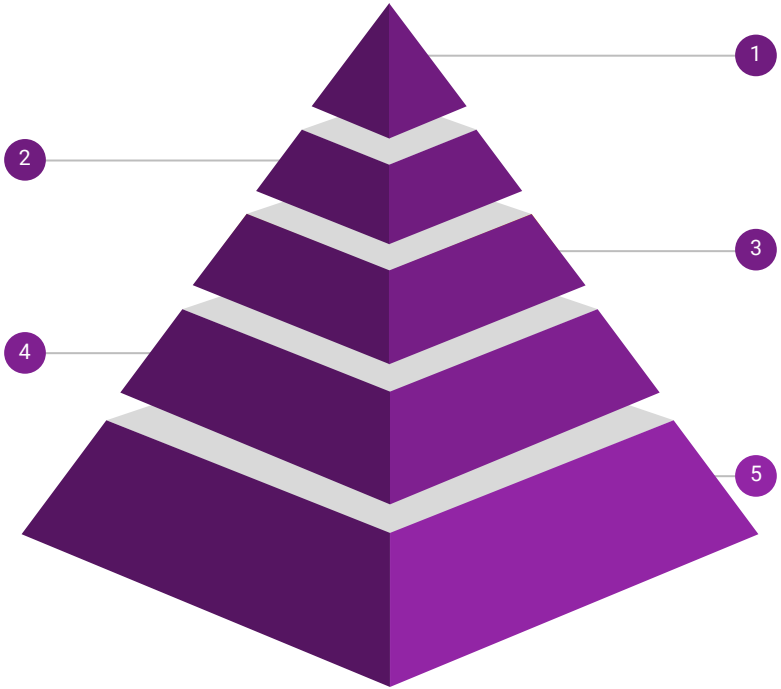
(SRE) Management 2.0

Write Code

Unless you like rust. Not the programming language; the other rust. You will also be so much more empathetic to your team's needs

Continuous Performance/People Management using tools

Data collection and analysis will help. Also Hiring.



Manager/Skip/Corp/World Alignment

Don't be oblivious. Align with your Boss, Boss' boss, the corp and the world.

Evangelize/Manage your team's products/tools/Projects

If you are not running, marketing and selling all the good stuff your team does, you are not doing your job well. Also, bring fresh ideas in.

Review your Teams' Code

You are more prepared for a good rep at the table during promo.You also stay sharp and contribute to the strategy.

Also, flatter orgs. Please. A clueless VP/Dir with 2 direct reports isn't going to cut it anymore.

Practical_solution #4: **Next-Gen SRE Tooling for the Enterprise**

$XM in Potential Savings; -50% in Toil

ATTR_1

Observability & Analysis of Unstructured data

ATTR_2

Observability of your Platform

ATTR_3

Opinionated Domain Cadence

ATTR_4

Human-Centric Design & Human factors

ATTR_5

(Async) Workflow Orchestration

ATTR_6

NLTs & Knowledge Graphs

Practical_solution #4: **Next-Gen SRE Tooling for the Enterprise**

+50% Productivity; -60% Stress

ATTR_7

NLP-Based Design

ATTR_8

Scalable, Multiplicative Actions.

ATTR_9

"Collaborative Interplay and Synchronization" *

ATTR_10

Context Modeling & Situational Awareness

ATTR_11

Asset/Attribute Correlation Mining & Intelligence

ATTR_12

Inclusive Postmortems/PIRs

* - source:

http://rave.ohiolink.edu/etdc/view?acc_num=osu1593661547087969

Future of SRE

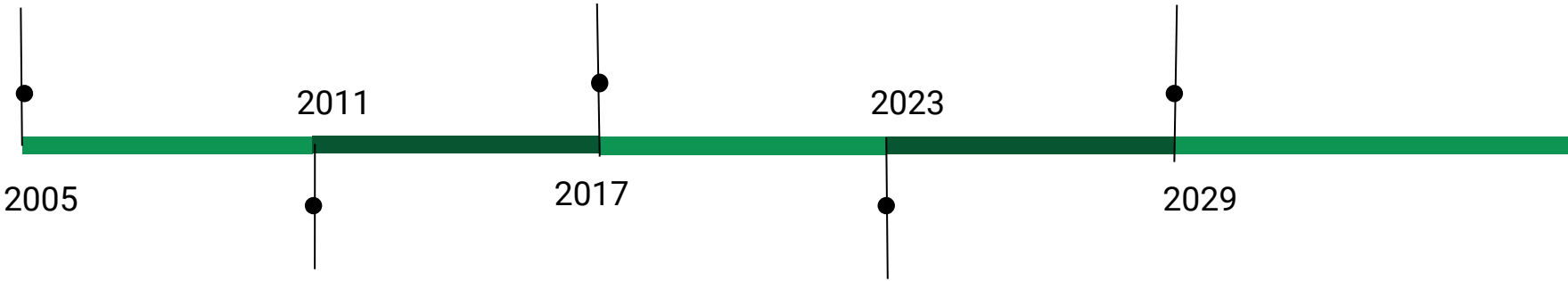
Servers
Code: Mostly concrete
Arch: Monolith
SRE: On Call

Cloud
Code: Abstractions Begin
Arch: Monolith
SRE: On call and/or embedded

(Micro) Services / K8s / Docker
Code: Abstractions Overload
Arch: (Micro) Services
SRE: Several flavours w/ multiple layers

Generative AI: ChatGPT, Etc.
Code: Abstractions Max
Arch: (Micro) Services
SRE: Define what SRE is for your org.

Generative AI -> Orchestrator AI -> Full-Stack AI
Code: AI with a persistence layer.
Arch: Full-stack AI
SRE: Everybody is a platform engineer. No more Dev or Ops delineation.



One last question

Whose job will it be to fix things when AI becomes unreliable or has an outage?

One last question

Whose job will it be to fix things when AI becomes unreliable or has an outage?

It's going to be the job of the SRE to fix things.

But, we need the right skills: "Platform Competence"

My attempt to contribute to the future of our industry

Building the Next-gen SRE/Platform Tooling

UnStruct

AI-Infused Process Orchestration & Observability for Incident Management.

I am looking to onboard 

Customers.

Q&A

Key takeaways Recap:

1. **Define what SRE is -- to you; not what it meant for Google.**
2. **"Platform Competence"** and professions like SRE/Platform Eng potentially becoming even more valuable with the rise of AI. Prereq is coding.
3. **Next-Gen, Human-Centric (SRE/Platform) Tooling** that can dramatically increase situational awareness (everything from detection, diagnosis, toil tracking, analysis, repair, escalations, etc.).
4. **Management/Leadership <=> IC Alignment via Code**; more leaders at the front lines; (re)define what it means to a manager/leader in 2023 and beyond.
5. If you haven't already done so, learn to **leverage AI to advance the profession.**