The Human Factor in SRE

Unveiling the root cause! SCORE !

Author

EHSAN KHODADADI

Date/Time

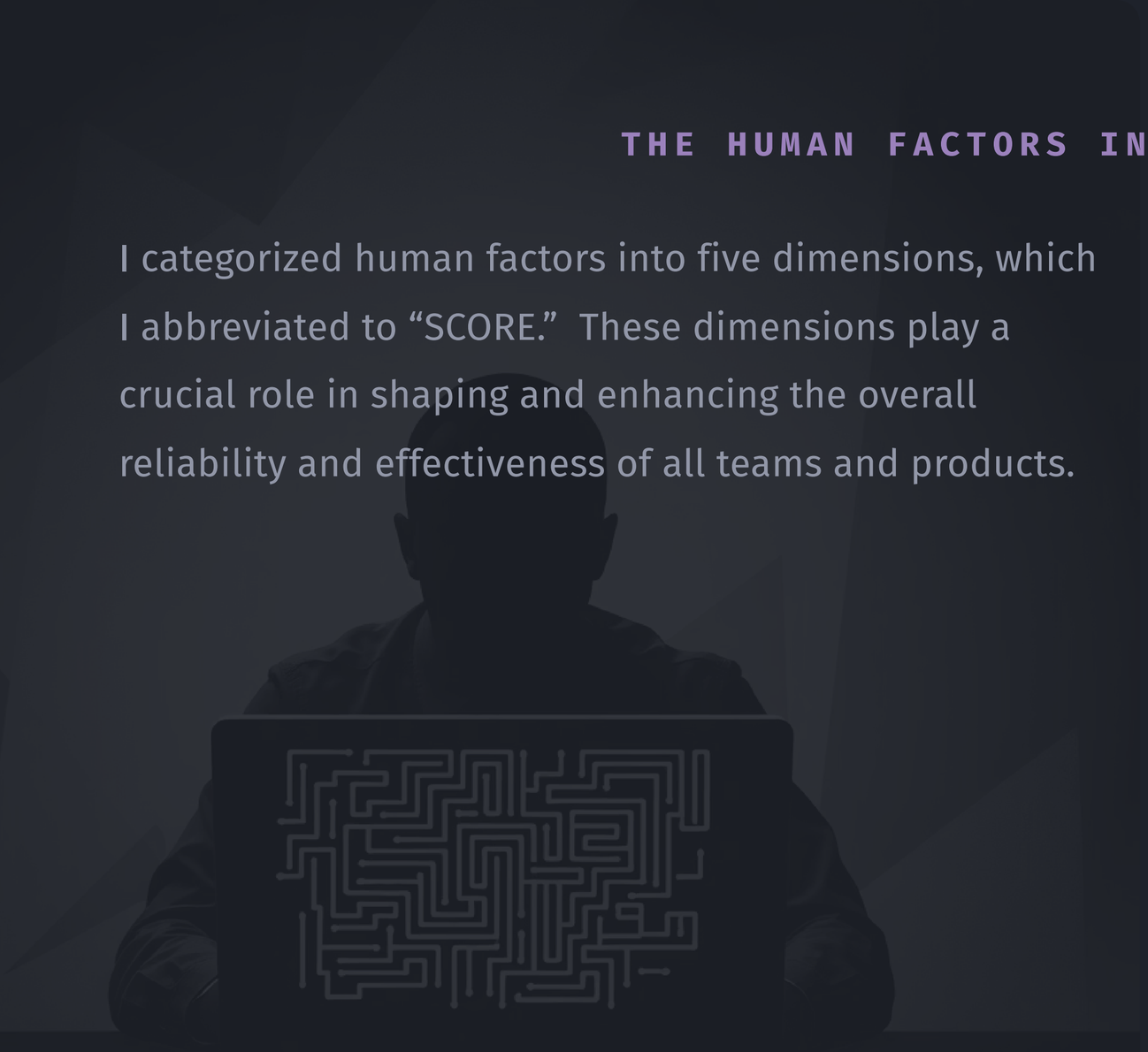
1736964928

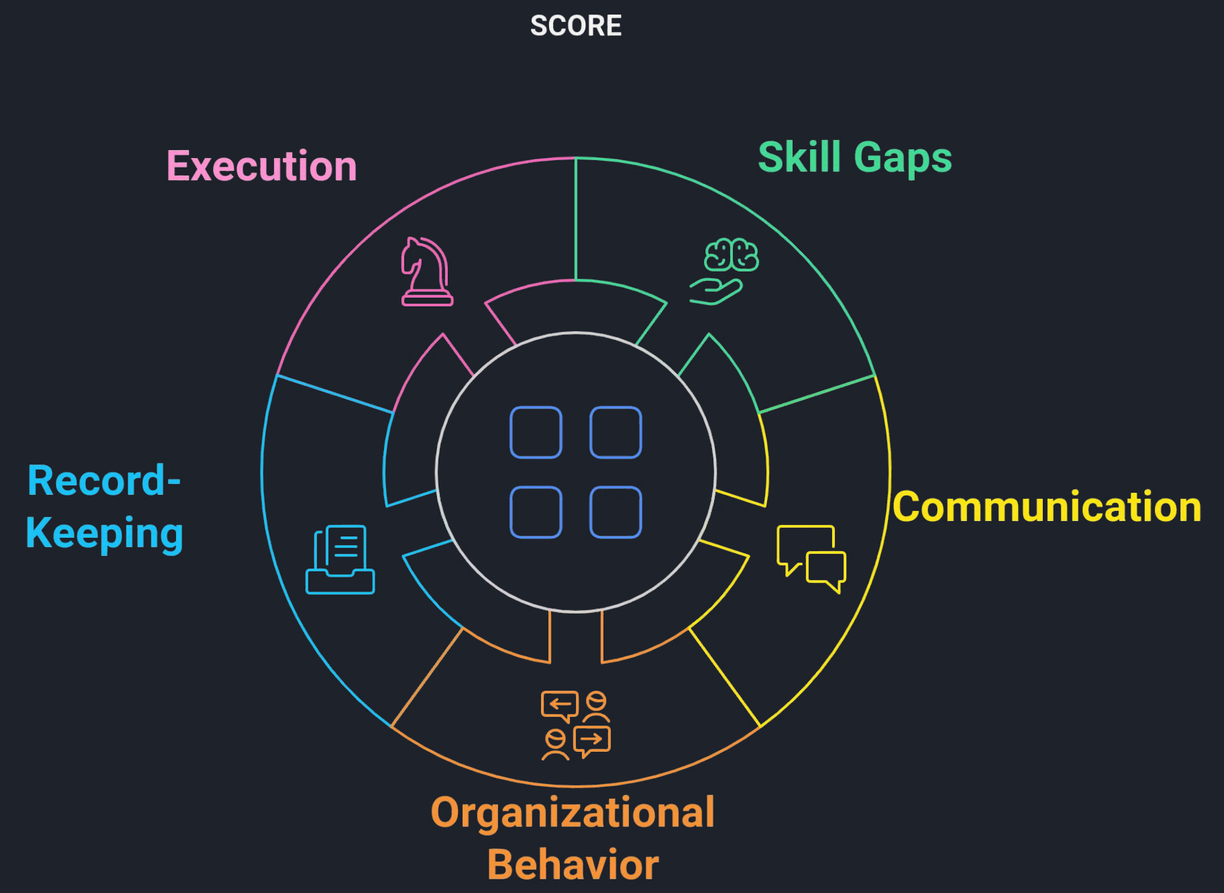
Who Am I? Why This?

```yaml
apiVersion: sre.google/v2
kind: speaker
metadata:
  name: ehsan-profile
  location: "SRE NL Meetup @Xebia"
  labels:
    expertise: "SRE"
    Born_and_raised: "Iran"
spec:
  summary:
    professional_experience: "Nearly 8 years in SRE, 20 years in computer engineering"
    key_achievements:
      - "Worked as an SRE team lead, forming SRE teams from scratch, DevOps/Platform teams"
      - "Enhanced operational reliability and observability"
    current_focus: "Optimizing observability solutions and contributing to the SRE community"
    personal_interests:
      hobbies:
        - "Plane spotting"
        - "Cooking"
        - "Exploring IoT innovations"
  why:
    bold_conclusion: "in 90% of cases, people are the root cause of issues"
```

THE HUMAN FACTORS IN SYSTEM RELIABILITY

I categorized human factors into five dimensions, which I abbreviated to “SCORE.” These dimensions play a crucial role in shaping and enhancing the overall reliability and effectiveness of all teams and products.





SCORE!

What is wrong with the culture?

Skill gaps

"it was working fine during testing, but I didn’t anticipate the high-traffic scenario.", "Circuit Breaker!? what the heck is that?"

Communication

“The dependencies have been changed and no one acted in time to update the code”.

Organizational behavior

“ We don’t know what was the actual root cause of the issue, we restarted our containers and everything works now”

Record-keeping (Documentation)

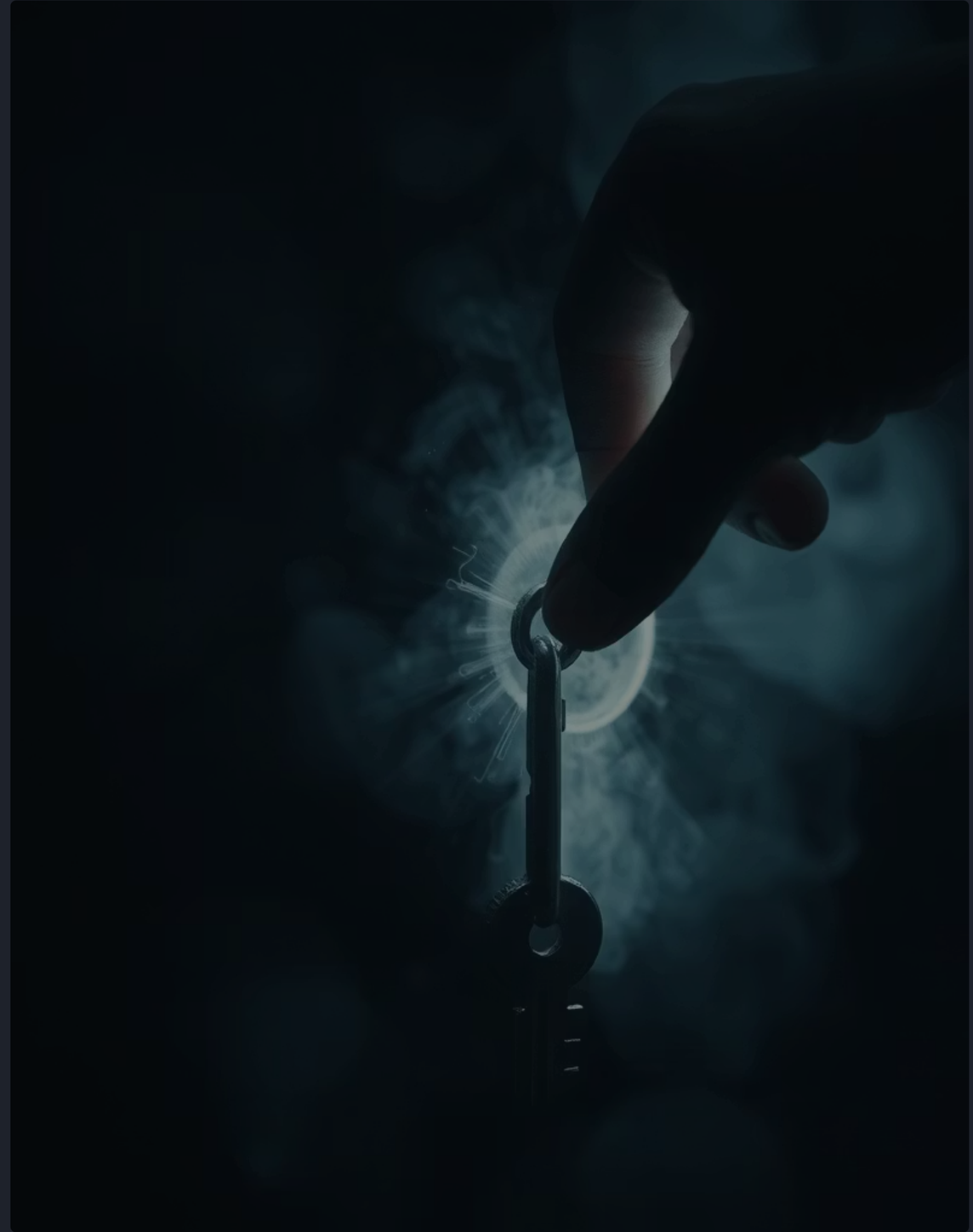
"The documentation didn’t cover that specific scenario."
"oh, I used another document, I didn't know there was another one"

Execution

“We had to implement the change in production without testing it in DTA because our regressions differ from those in production”

What can SRE do?

Effective SRE can revolutionize the whole company by planting the seeds that help every engineering team bloom to their full potential.



Organizational behavior

Top-down approach

Bottom-up approach

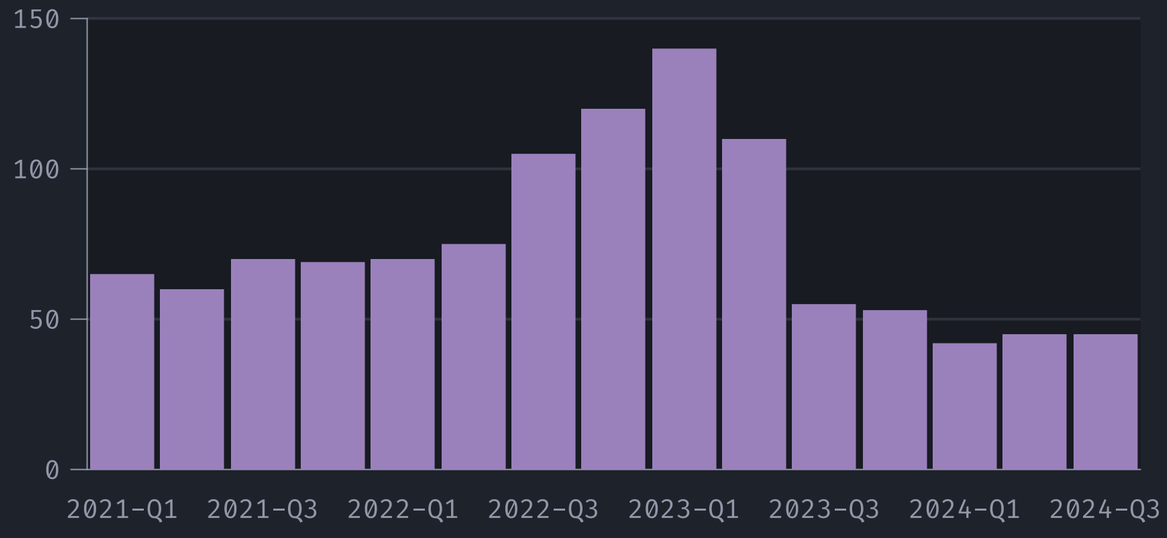
Incident management learnings

An incident is an investment where you have involuntarily paid the majority of the cost upfront.

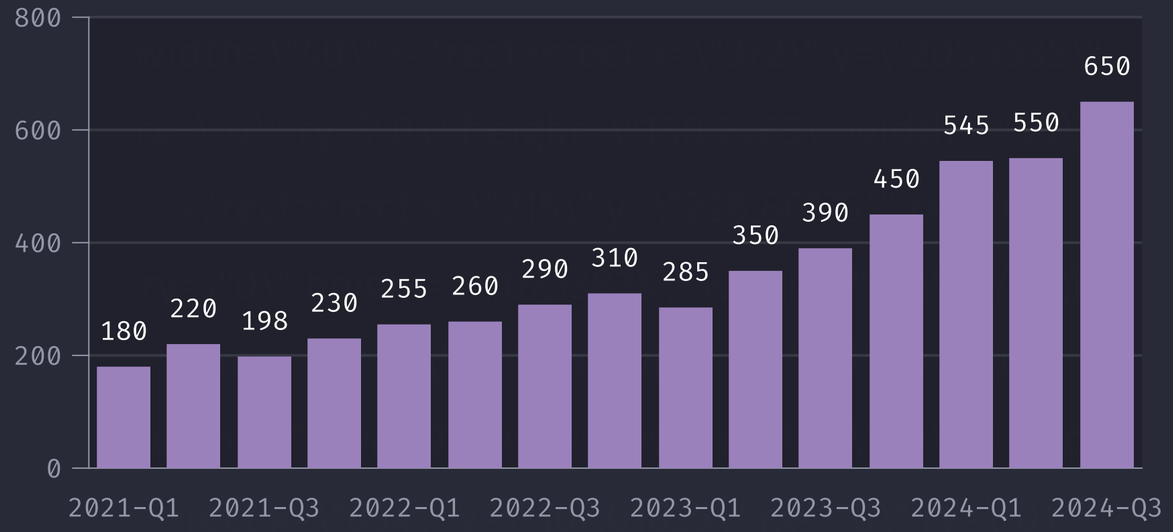
John Allspaw!

- Daily/weekly/monthly incident reviews and changes/updates accordingly
- blameless post-mortems and follow-ups
- Chaos engineering
- After incident knowledge sharing
- Page whoever needed
- Incident Gamification
- run-books and observability
- SRE reports
- Data driven decision making

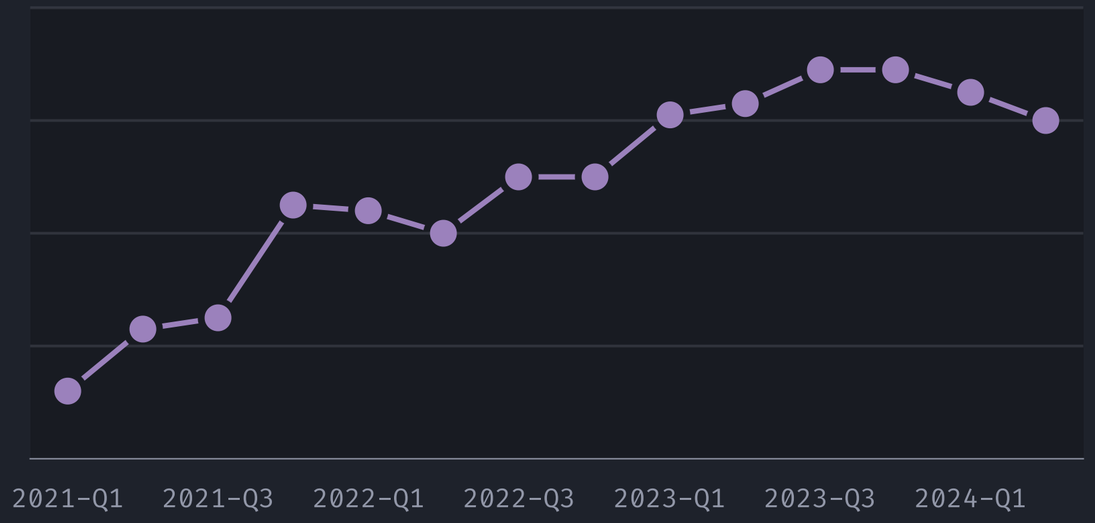
- An increase in the number of incidents is not bad
- Increased complexity results in more incident
- incident per number of users/engineers/revenue
- Tech Dept and complexity
- Daily/weekly reviews are the best way to learn
- Changes should be recorded, followed to the last step



Number of total p1-3 incidents/Q



Number of total incidents/Q



Number of total commits

COMPLEXITY AND TECHNICAL DEPT

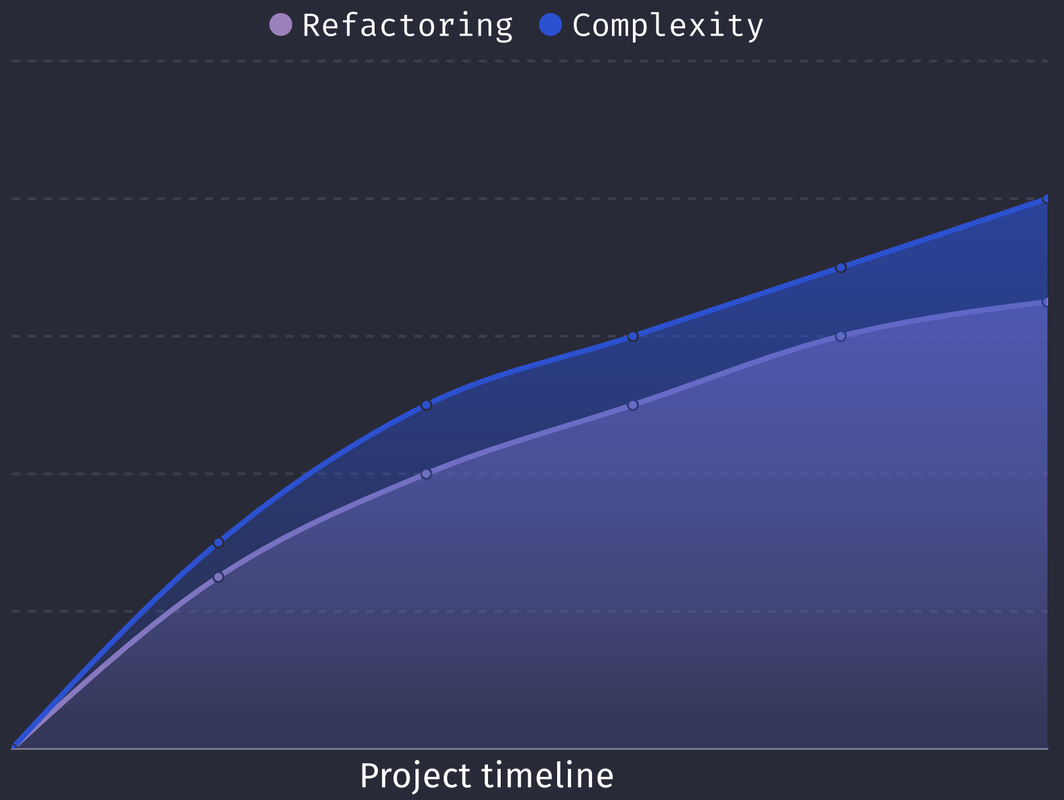
Technical debt is the gap between the world as it was when the code was written and the world as it is today.

20-80 rule

- ☑ 20% dealing with Technical debt
- ☑ 80% dealing with releases

Embrace complexity

Complexity is an inevitable indicator of growth and progress. It should be managed effectively to avoid increasing technical debt. Refactoring is an ongoing process!



Game Days

GAME DAYS ARE A METHOD OF INCIDENT PRACTICES

Game days should be taken seriously

Design practical Scenarios and make sure that the goals are met

Test what you think you and the system can handle,

ensuring that everything works as expected

Controlled environments for testing

Clarify the expected outcomes

OBJECTIVES OF GAME DAYS

- Identify related weaknesses
- Improve Incident Response
- Validate observability
- Testing run books
- Testing downstream and upstream services resiliency
- Bridge the Skill gaps

How much communication is good enough?

```
def how_much_communication_is_enough():
    # Check if you're communicating enough
    if not feeling_confident_about_communication():
        print("Am I communicating enough?")
        print("It's okay to reflect, but don't let doubt hold you back.")
    # Check for over-communication
    if messages_are_repetitive() or audience_feels_overwhelmed():
        print("Am I over-communicating?")
        print("It can be hard to know the balance, but don't worry.")
    # Ensure acknowledgment and understanding
    if not receiving_acknowledgment():
        print("Am I being acknowledged?")
        print("Seek confirmation to ensure you're heard and followed.")
```

The key is to know your audiences! More egress communication makes more ingress!

COMMUNICATION CHANNELS! AND BACKUP CHANNELS!! AND BACKUPS FOR THOSE BACKUP CHANNELS!!!

Documentation!

- Write documents with no pre-assumptions
- Write easy-to-read docs and add enough code samples
- Archive/delete outdated docs
- Write comprehensively
- A diagram says way more than just text (A diagram is worth a thousand words)
- Add links to other pages
- Use good use of tags (Ensure documentation is discoverable)
- know your target readers
- Use AI as much as you can
- Make diagrams comprehensive enough
- Create templates for consistency
- Choose searchable formats
- Establish regular review cycles
- Step-by-step procedures for How tos

Thank you!
Don’t worry, every good SRE has brought down production at least once!

Lessons Learned from Twenty Years of SRE

Q&A

https://www.linkedin.com/in/ehsankhodadadi/

ehsan@ehkh.nl
ehsan.khodadadi@leaseplan.com