SKILupDAYS

DevOps Institute

Evolving with SRE – The Game Plan

Santanoo Bhattacharjee
Solutions Expert – DevSecOps & Hybrid Cloud | Accenture Advanced Tech Centers
DevOps Institute Ambassador
/NexGenOps

The Outline

Get the basics aligned

- Why do we need to evolve?
- Are to even ready to evolve?
- How are we different in adopting this?

Visualizing the transformation

- What all do we need?
- Why do we need something new?
- How will the dots connect?

The Game Plan

- Refining the Metrices
- Mapping Capabilities
- Identifying the Velocity
- Scoping the lens of Release Roadmap

The Road Ahead

- Have we structurally aligned the goal with the journey?
- Are we really disruptors or evolvers?
- What lies ahead…

Aligning the Basics

- Why do we need to evolve?
- Are to even ready to evolve?
- How are we different in adopting this?

Are you aiming to scale fast & reliably?

Are we implementing changes, frequently or occasionally?

Are we measuring our delivered software behaviors?

Do we have a short feedback loop?

We aren't that agile! Do we really need all these?

We have a dedicated team of experts, all the time. Are we good?

Accelerates Delivery

DevOps

- Reduce Organizational Silos
- Implement gradual changes
- Leverage Automation
- Ecosystem of integrations
- Accept failure as normal

vs

Site Reliability Engineering

- Share Ownership
- Evolve by measurement
- Reduce toil, improve efficiency
- Evolving architecture
- Failure is good (if it is detected & fixed)

Stabilizes Delivery

The Foundational Principles

Common Goal – Create a highly scalable & reliable ecosystem of delivering & maintaining software

Scaling up is chaotic, it needs standardization, forming a process

Visualizing the Need

- What all do we need?
- Why do we need something new?
- How will the dots connect?

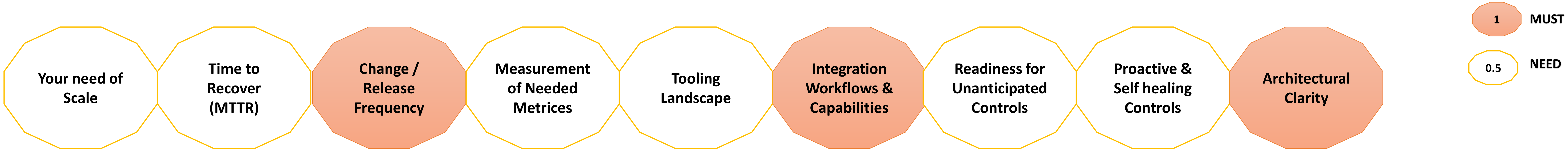
Buzz Words / Abstract Definitions

Reduce Toil / Cost of Failure / Shared Responsibility / Automate / Managed Service Alignment / Operation Driven Development etc. etc.

Key Point

"you built it, now you run it"

"build automation to reduce toil, increase observability, and improve reliability of the systems "



Your need of Scale

Time to Recover (MTTR)

Change / Release Frequency

Measurement of Needed Metrices

Tooling Landscape

Integration Workflows & Capabilities

Readiness for Unanticipated Controls

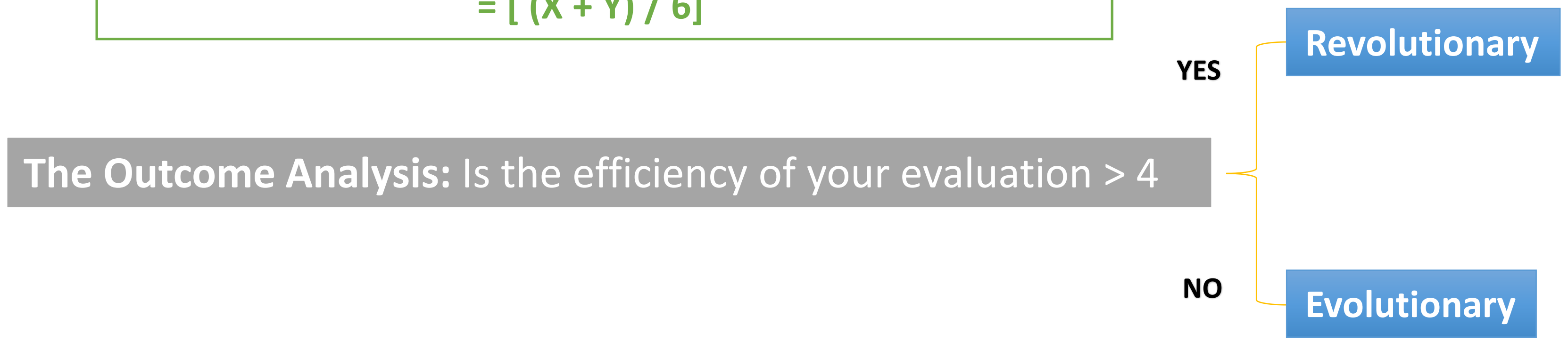
Proactive & Self healing Controls

Architectural Clarity

1 MUST

0.5 NEED

EFFICIENCY SCORING = [(Must + Need) / Total Coefficient]
= [(X + Y) / 6]



The Outcome Analysis: Is the efficiency of your evaluation > 4

YES – **Revolutionary**

NO – **Evolutionary**

Evolved from Delta 4 Theory by Kunal Shah, Founder @ CRED

You can put all your money to build the best app in the world, but if it is not up – it's all in vain!

The Game Plan

- Refining the Metrices
- Mapping Capabilities
- Identifying the Velocity
- Scoping the lens of Release Roadmap

1 The Value Stream

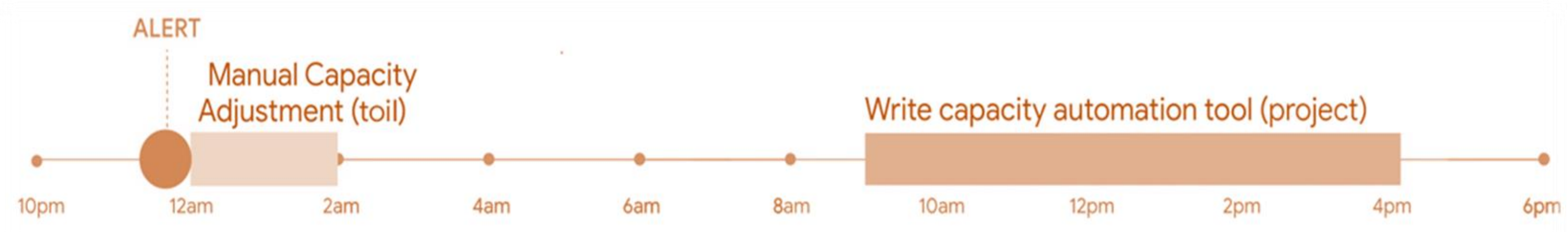
1. End – End Flow Mapping
2. Positioning your DevOps Ecosystem
3. Identifying Value Creation, Leakage & CSI
4. Positioning your SRE as a Capability

2 Patterns & Anti-Patterns

1. Islands of Automation
2. Tools are just enablers, accept that
3. Component level integration vs Service level integration
4. Clarity on workflows of products

3 Scoping through right lenses

1. Re-defining your SLI, SLO, SLAs
2. Error Budgets - Balancing innovation & stability
3. MTTR – "Quantitative Measurements"
4. Toil Factors – automatable, daily vs worthwhile
5. Make Reliability a Key Pillar



4 Observability as a Capability

1. Scoping The "Known Unknowns"
2. Data Channeling through Signal Synthesis
3. Tools promise a lot – Align it (system & human errors)
4. Simulate scenarios & build root cause channels
5. Structured Logs vs Metrices vs Traces

5 Collaboration is the key

1. Instead of WHO, lets focus on WHAT is getting contributed
2. Divisional vs Central Setup
3. Focus on data driven feedback loop

"For any transformation, The whole contextual philosophy is different company to company"

The Road Ahead

- Have we structurally aligned the goal with the journey?
- Are we really disruptors or evolvers?
- What lies ahead…

> 50 %

Time & effort of SREs should be channelized into capability development such as enhancing observability, predictive healing, automation etc.

> (Tech Stack) > (Complexity Cost)

While these technologies solve some problems, they create an additional complexity cost. The developer would need to understand all those technologies and services in addition to the core technologies (e.g., languages) the application uses

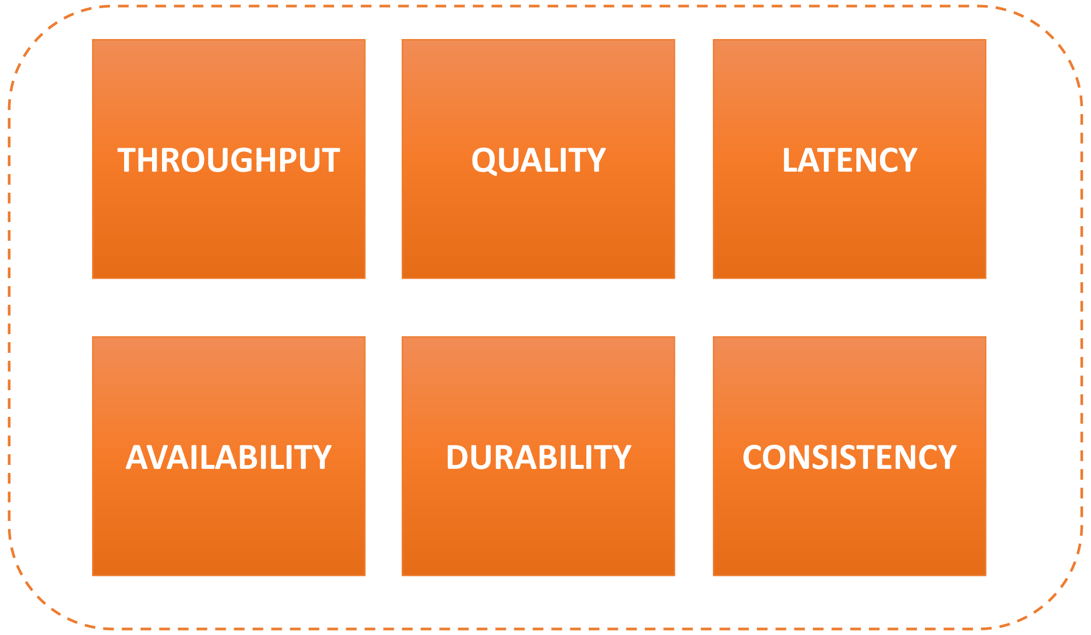
$Rewards$

Start Incentivizing. Both the SREs and developers have a strong incentive to work together to minimize the number of errors. Create a self-policing system where developers get rewarded with more teammates for writing better performing code

“SLIs drive SLOs which inform SLAs”

Define SLA with Error Budgets with data backed evidences, Have clear numerical indicators.
Note - If you’re trying to increase your SLO’s way too much, you end to delaying your release to features

RELIABILITY



THROUGHPUT

QUALITY

LATENCY

AVAILABILITY

DURABILITY

CONSISTENCY

Food for Thought

“Do not emulate Google, They have a completely different structural case of having SRE in the first place. You will need to build reliability engineering constructs to refine your needs!”

For all of us, the reliability engineers – a key question should always be clear "We train & prepare ourselves for years just to be ready to fight something which has the power to lead to a chaos in minutes".
The key lies in, "how much ready will we ever be!"

Thank You