Want to Solve Over-Monitoring and Alert Fatigue? Create the Right Incentives!

Kishore Jalleda (@KishoreJalleda)
Sr. Director, Production Engineering, Yahoo.

Over-monitoring & Alert fatigue in ICUs

"Research has shown that 72%–99% of ECG monitor alarms are false or clinically insignificant.

"Important clinical events may be missed amid the cacophony."

Source:
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4926996/

Patient Deaths from Alert Fatigue

"For instance, in 2010 at a Massachusetts hospital, a patient death was directly linked to telemetry monitoring after alarms signaling a critical event went unnoticed by 10 nurses."

Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4926996/

At 99% false alarm rate, service is "best-effort".

"ECRI Institute has placed alarm hazards in first or second place on their Top 10 Health Technology Hazards annual list since 2007"

"The Joint Commission has made the task of improving the safety of clinical alarm systems a National Patient Safety Goal for 2014"

Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4926996/

Lessons learned from healthcare

More is not better; must use it judiciously & wisely.

More monitoring = More money

Monitoring w/o purpose = Wasted Alarms

How can we solve this problem –in Tech– where it is equally rampant?

Let’s look at an example - Zynga

Zynga Stats*

Hybrid Cloud Pioneer (AWS + Zcloud)

“(Micro)services”

40k+ servers

125+ million MAUs

*** From 2013**

Alerts

100k+/month across 25+ studios

40+ person SRE team in three locations
(Tier 1, Tier 2 & Tier 3)

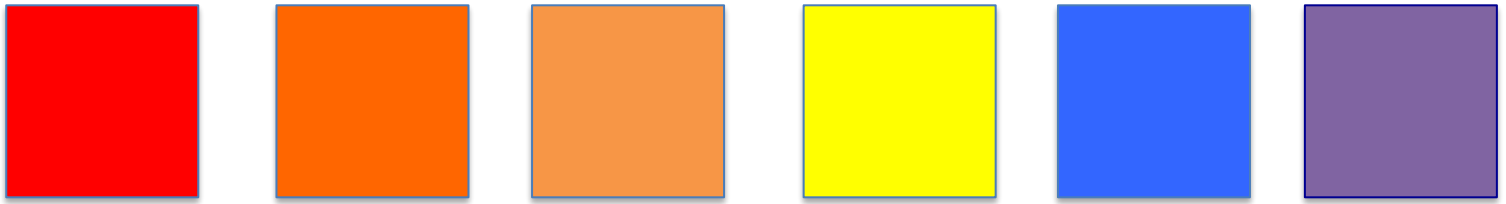
Alerts -> Tier 1 -> Tier 2 -> (Tier 3) -> Dev

SRE was “Best-effort”

NOC (tier 1) had 30+ TVs

SRE was “Best-effort”

Alerts color coded – based on priority



SRE was “Best-effort”

Some teams given priority on revenue their products made (or whoever made the most noise).

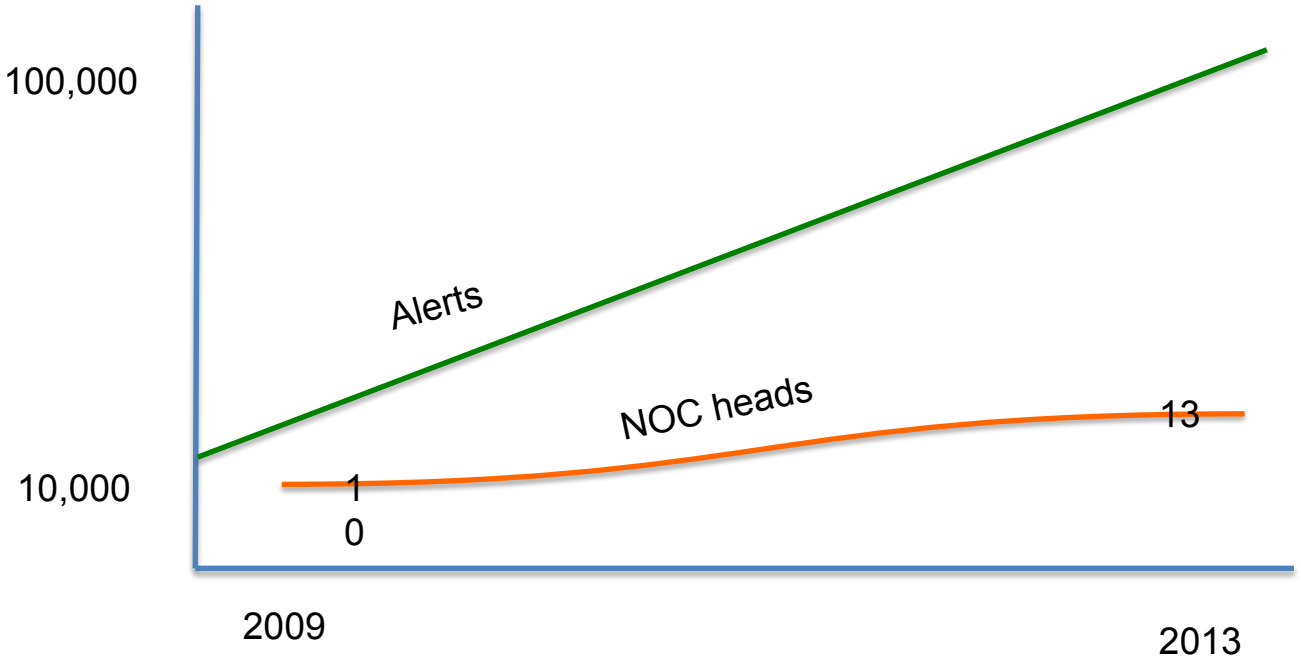
As a result

Frequent Outages ☹

Customers suffered ☹

Every day was scary; filled with anxiety. That dreaded escalation call ☹

Adding heads is the not the answer; that can't scale.

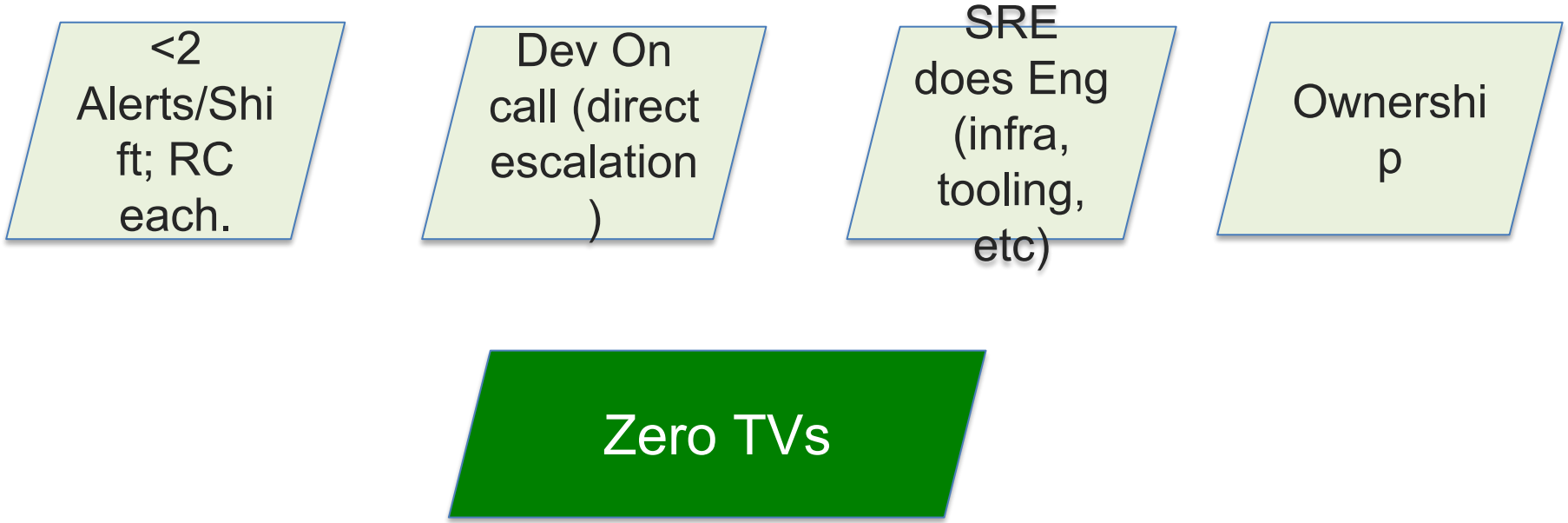


Filtering noise by color coding alerts based on priority is not the answer; soon, you will run out of colors (like we did).

More tools (to filter noise) is
not the answer

So, what is the answer to this problem?

Before you solve the problem; first, have a Vision.



Then… Socialize your Vision

Get feedback

Some laughed; most were skeptical.

Conversation with a Dev team:

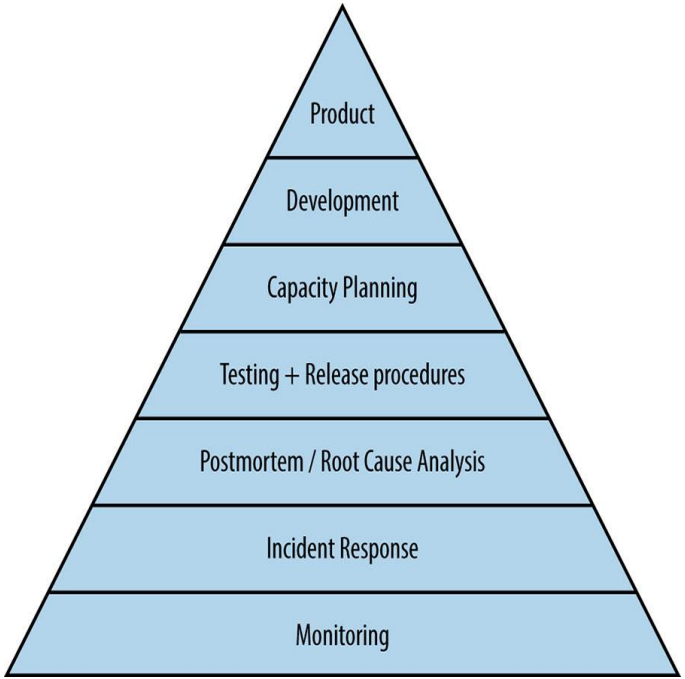
Dude, stop spamming; nagios is (all) red. Can we talk?

Response from the Dev team:

Sorry, I know you are the new guy; but, can you please have your team respond to alerts in a timely fashion first?

…I was like ☹

My Teams on the other hand:



Were trying to do too much; were stretched too thin.

Source: *https://landing.google.com/sre/book/chapters/part3.html*

Engage. Persist.

Made some progress, but things were not going too far; wasn't sure exactly how to blaze the trail.

Then, one day, there was a massive outage. There there were more outages; same root cause.

During one of the outages, I had this Light Bulb moment. A brilliant idea.

I decided to deny (SRE) support.

Based on Alert budgets

Go over & you are kicked out (temporarily). I.e. no SRE support.

Stay within budgets & you are rewarded with world-class SRE support.

But, Saying “No” is hard; but powerful.

“The most important skill any leader — any person, really — can learn is how and when to say “no.””

Source: https://hbr.org/2017/06/help-your-team-stop-overcommitting-by-empowering-them-to-say-no

Transformation change rule #1: Leverage Outages

I used it to drive reliability. Conduct Postmortems. Send weekly follow ups. Build Credibility. Show that you care (a damn).

Do enough so people take you (and your vision) seriously.

The aha moment

As I was meeting more teams, I found that all of us wanted the volume to go low; just that no one was looking at this holistically.

SRE was in that unique position to do so being a horizontal function.

Rule #2: Find your Allies

Who are (radially) aligned with you. In my case, Zynga's CTO.

There are always going to be some; you just have to find them.

Rule #3: Call your Boss and tell him this:

“This is it; I am going to be making some pretty dramatic changes - we should expect some major shake up because of this".

Rule #4: Get Buy-in from Senior Leaders

Leverage partnership & collaboration opportunities. Be visible. Campaign.

Go prepared to the meeting; be ready to speak about vision succinctly.

Said “no” to my team

“Baby steps, please! Let’s focus on one thing (monitoring); execute on that one thing; prove that we can deliver; then, we can then take on more.”

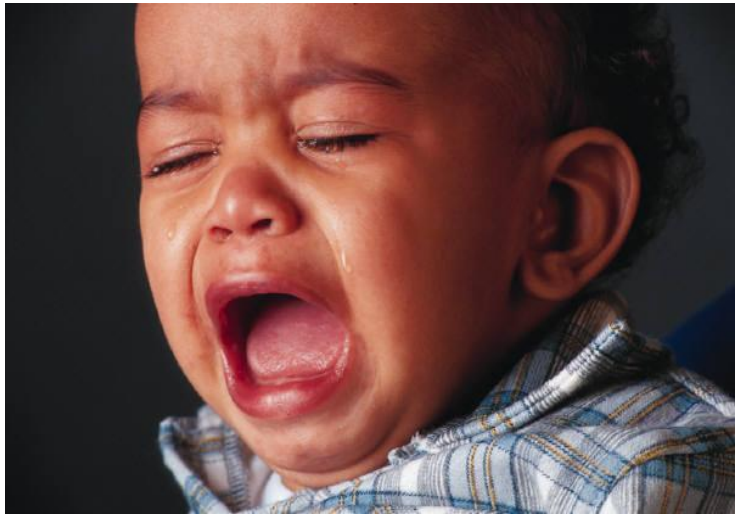
Said “no” to the Dev teams

“Sorry, but, not every team deserves SRE support; it must be earned”

Response from (some) Dev teams:

Are you f***** kidding me?

- “We pay your salaries”
- “Where can hire SREs now?”



Yes, there were (lot of) escalations to my Boss; and his boss -- but, remember rules #3 and #4?

Response from my team:

“What about our job security?”

After all, I was proposing that we take some responsibilities away (temporarily).

Some threatened to quit

Eventually, they did (you should expect that s***)

Some, you have to let go; or move them to a different role.

Do not forget the power of "No"

There will be conflict.

If you can stand firm, learn to say "no", the outcomes can be pretty awesome!

Also, it's not enough if you --as the leader-- say "no"; you must empower all of your teams to also say "no".

My promise to my team

Higher-value-add work: writing software; infra; tooling; etc.

Trust me, you can’t possibly automated yourselves out of your job.

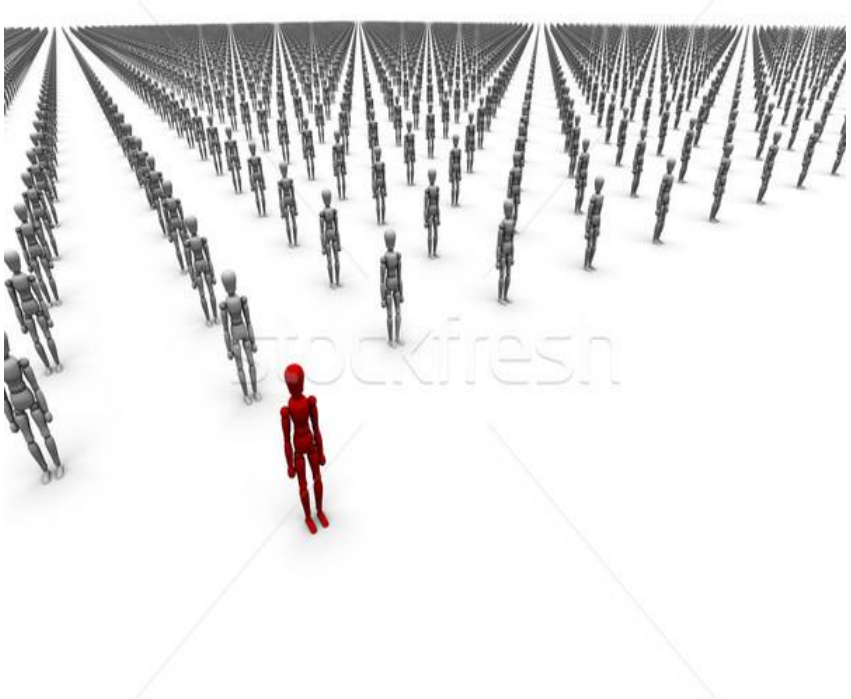
“Clean Room” Contracts Established

Base lined alerts; came up with targets

Gave teams time to clean up before launch.

Became a Corp initiative

When senior leaders believe in your vision, you have a greater chance of success.



Great, but how do you go about actually reducing alert noise?

First, define an Alert

- All alerts are actionable; must need immediate (<15 min) response

- Require human intelligence to fix

Reduce ALERT noise – Process/Culture

- Daily incident/alert reviews
- Weekly reports
- Public shaming ☺
- Peer Pressure ☺
- Budgets /incentives
- Ownership
- Tickets vs Alerts
- Dev On Call

Reduce ALERT noise - Tools

- Alert aggregation and grouping
- Auto Remediation
- Logs vs Tickets vs Alerts
- Symptoms vs RC monitoring
- Avoid tight coupling with abstractions.

Framework for “clean room” was built with Transparency & Accountability in mind

Every alert’s lifecycle must be tracked; everything must be instrumented.

Baby Steps - Pilot

Leverage those allies; those relationships.

Show small wins; one win at a time.

Momentum will pick up.

Launch

Went well. Few teams were kicked out. Ruthlessly (power of “no”); some were given a bit more time (e.g. attrition in the team)

Make sure whole company knew about the launch. Communication will never be enough. Trust me.

“Clean Room” was a success:

- 90% drop in false alarms
- 5 min SRE response times
- SRE credibility was (almost) back
- Teams wanted us to take on more
- Uptime went up by a whole nine; customers happier.

Incentives can shape Culture

All I did was to align incentives by saying “no”; after all, the root cause was that the incentives were not aligned.

Closing thoughts

Dev & Ops: do you have it backwards?

- Pushing code you do not own?

- Responding to alerts for products you don’t own?

- Testing & debugging code you don’t own?

- Writing tests for code you do not own?

- Etc.

Ask yourself, are you really adding value?

“The misuse of talent in large organizations is rampant today”

“Without the ability to say “no” to low-level tasks in order to say “yes” to groundbreaking ones, people stop innovating”.

Source: https://hbr.org/2017/06/help-your-team-stop-overcommitting-by-empowering-them-to-say-no

A better model (call it “DevOps” or whatever):

Core Dev Teams own build, test, deploy, monitor, on call, debugging, incident response, capacity, Postmortems, etc.

Non-core Dev & Ops Teams own infrastructure, automation, network, dev tools, expert services, etc.

Do yourself a favor

Read this awesome post by an SRE at Google, JBD, @rakyll … https://medium.com/@rakyll/the-sre-model-6e19376ef986

Please!

Reflect; soul search; ask tough questions

How is my team providing value?

Why does my team exist?

What is the right engagement model?

Is my team detracting value?

Am I adding unnecessary abstractions?

Question s*** like this:

"We have been told that teams X, Y are tiers 1 & 2 for the whole company"

"We resolve 80% of 50,000 alerts. I am like, "that's bull****".

Our responsibility to define the future.

Do the right thing:

- Enable a Culture of Ownership.
- Engineer Automated & Agile Processes (Iterative & Experimental)
- Develop Self-Serve & (Re)usable Tools.

Yes, there will be exceptions. But handful. Cash cows, for example.

Transformational change - First 90 days

No big decisions for the first 60-90 days - help someone remind you (your boss?)

Don't come across as a jerk - even if you did not intend it that way.

Execute on one thing; build credibility; go for the fancy s*** later.

Aligning incentives.

Learn to say “no”. Persist.

Leverage top-down corp initiatives; they make it easier

Socialize your vision – relentlessly; get people excited.

Walk the talk: be on the front lines; engage with customers; be on call.

Go to people's desks and tap them on their shoulder; hold them accountable.

Learn all the acronyms.

Are you ready to say “no”?

Thank you!

(Questions?)

Twitter: **@KishoreJalleda**