Summary:
For over two decades, Google has used many variations of team organization to drive its reliability goals. We've found that reliability must both be a first-class goal for the organization and a key component of its service architecture.

In this article, we focus on the organization and its ethos that contribute to the reliability that the organization is capable of sustaining. We describe stages of reliability maturity that an engineering organization can transverse; how to figure out *where* in the continuum of maturity your organization currently falls; provide some ideas or experiments that can be used to improve the reliability maturity of the organization, and, most importantly, provide some thoughts on how to determine the reliability maturity phase the organization requires. Not all teams need to achieve or maintain the same level of maturity, and knowing what best suits your organization is critically important.

Reliability Maturity

Over time, we've found that reliability must not only be a key component of its service architecture but more importantly, a first-class goal for the organization—similar to feature quality or feature performance. There are multiple books on strategies for developing the service architecture which underpins a reliable service, for example, Building Secure and Reliable Systems (Adkins [1]). Nevertheless, having a reliable service architecture does not fully guarantee reliability. It is critical that the technology solution is complemented with a robust culture, advocacy, and sponsorship for reliability at all levels.

Our thoughts on Reliability

Based on our combined experience as SRE and development leaders spanning critical products and core services at Google, multiple surveys, and the perspectives of cross-functional leaders, we learned that people generally pay more attention to reliable products rather than the culture and mindset of an organization which created those reliable products in the first place. We believe that the overall reliability of a product is a property of the architecture of its system, processes, culture, **and** the mindset of the product team or organization that built it. In other words, reliability should be woven into the fabric of the organization.

At Google, we've developed a terminology to categorize and describe an organization's reliability mindset. These categories exist on a continuum. Not all products and users demand or will pay for a very high reliability level, though some products, due to the nature of the application, demand very high levels of reliability. The investment that the organization makes in reliability comes at a cost, and it should be understood if the return on investment (ROI) supports the level of reliability. For example, Google Search demands a robust reliability culture, and continues to invest to maintain this culture.

Twenty five teams were surveyed at Google as to their state and this was the distribution of those teams; this is not an exhaustive study of all of Google. Based on our observations, there are five basic stages of organizational reliability. These stages are based on an organizational maturity model of:

- absent
- reactive
- proactive
- strategic
- visionary

These phases describe the mindset of an organization at a point in time, and each one has a set of attributes that can be detected. The attributes of a reliability maturity model were developed to conform to the accepted structure of the *absent* to *visionary* model of maturity. These attributes were developed while observing several years of organizations moving in and out of the proactive phase at Google. Significance was placed on the people and culture, since this is how the proactive phase maintained its stickiness.

The attributes of the organization that allow this classification are:

- Operational processes - activities and metrics the organization takes to maintain production health;
- Risk management - evaluation, identification, and resolution of reliability risks;
- Productivity - impact of reliability efforts on developer velocity and feature launches;
- System complexity - the architecture plus dependency graph of a product/service;
- People - operating model, values, team ethos;
- Leadership - commitment of leadership across the organization to reliability (not in a siloed team);
- Visibility/Culture - acknowledgment of success/achievement for reliability.

The use of organizational maturity models is not unprecedented in the study of organizations focused on safety, such as aerospace, healthcare and civil engineering. For example, a framework that not only optimizes safety culture on construction sites, but also intrudes into the components of the organization is very desirable (Gellar 18-24, [2]) .

What this model is not

These organizational levels are meant to be fluid, and modified according to what makes sense to the organization. We highly recommend that the model does not become gamified, which drives leaders to inflate their levels or scores for the sake of appearances. Or said another way, we recommend avoiding implementation of Goodhart's law. Rather, the analysis of the current and desired phase should be done introspectively by the leaders of the organization to ensure that the analysis is accurate and proper steps can be taken to improve, if required.

Specifically, this model is *not* a prescriptive checklist of things to do that will improve reliability, nor mandated principles which every one should apply in the same fashion.

Who is this model for?

This model is for organization leaders who are looking to build, and more importantly, sustain a reliability mindset in their organizations. At Google, we discovered that short-term sprint efforts to improve reliability would temporarily improve the reliability of a product. However, the habits of the organization would quickly drive the product back into a position of unreliability. Reliability should be systemic across the organization, embraced in all processes, and celebrated.

In the context of this model, *organization* is defined as a group of cross-functional teams owning a product or service. For example, an organization might contain Product Management, Quality Assurance, Development, UX, Site Reliability Engineering, and Support. Finance should be a participant but is usually in a separate organization.

Where are you on the reliability spectrum?

It's very important to understand your organization does not necessarily need to be at the strategic or visionary level. There is a significant cost associated with moving from one level to another, and the cost to remain there is high. In our experience, being proactive is a healthy level to target and is ideal for most products.

To illustrate this point, here is a simple graph where various Google product teams are on the organizational reliability spectrum. As you can see, it produces a standard bell-curve distribution. While many of Google's product teams have a reactive or proactive reliability culture, most can be described as proactive. As the organizational leader, you must consciously decide to be at a certain phase based on product requirements and client expectations.

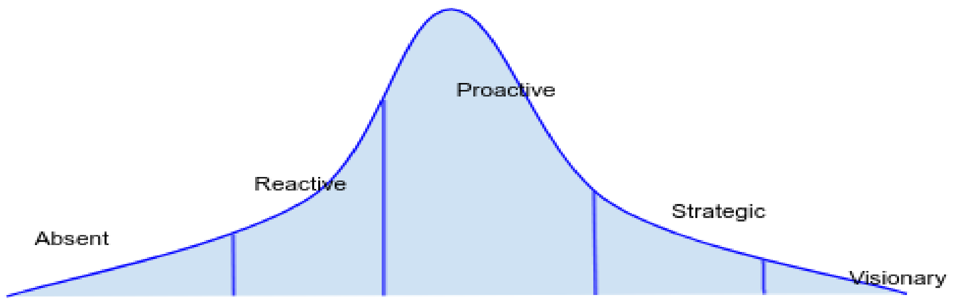

Figure 1 - The organizational reliability spectrum

Furthermore, it's common to have attributes across several levels. For example, an organization may be largely reactive with a few proactive attributes. Team culture waxes and wanes between levels, as it takes effort to maintain a strategic reliability culture. However, as more of the organization embraces, engages and emphasizes reliability as a key feature, the cost of maintenance decreases.

The key to success is making an honest assessment of what phase you're in, and then making a concerted effort to move to the phase that makes sense for your product. If your organization is in the absent or reactive phase, remember that many products in nascent stages of their life cycle may be comfortable there, in both the startup and long-term maintenance of a stable product.

The organizational reliability continuum

Below, we describe attributes that an organization would reflect, based on being in one of the levels (*absent*, *reactive*, *proactive*, *strategic* and *visionary*). In each phase, we also indicate practices that an organization would employ to move from phase to phase as required or desired.

The transitions from phase to phase are long, and maintaining a phase above the reactive phase takes energy and focus by the leadership team. These transition times vary by size and current culture of the organization, and its appetite for investment. Keep in mind that these are cultural changes for an organization, which are difficult. Additionally, it takes time to build the cultural muscle to easily maintain your phase. Do not underestimate the investment and time required to move to a higher phase. It is recommended that each phase is baked-in for some time before attempting to move to the next phase.

High-level Summary of the Model

Phase	***Absent***	***Reactive***	***Proactive***	***Strategic***
Summary	Reliability is a secondary consideration.	Response to known reliability issues/risks are tied to recent outages.	Potential reliability risks are identified and addressed.	Classes of risks are managed and architecturally addressed.
Description	Teams react or respond to specific issues/bugs or user reports when escalated from outside only.	Teams have some reliability related metrics defined and react to the metrics. Reliability work is prioritized when there are outages.	Teams/Org prioritize reliability along specific or localized dimensions based on risks and this is part of the normal operations/Objectives and Key Results (OKR) process of the organization.	Reliability is inherent and ingrained in how we design, operate and develop software as a cross-functional organization.
When it is appropriate to be at each phase	A product/project is in prototype phase.	A product/project is in pre-launch or stable maintenance phase.	Most services/products can be at this phase.	Services/Products that need very high availability to meet business critical needs.

Impact on Reliability of product/system	No expectations on Reliability.	Can sustain reasonable availability and can reach higher availability with heroic efforts by few individuals or spontaneous code yellows.	Can easily sustain high availability, achieved with a predictable reliability tax impacting developer velocity, efficiency and dedicated heads spent on reliability.	Can sustain very high availability while at the same time have efficient use of resources and optimal developer velocity, with minimal dedicated headcount on reliability. Reliability is systemic.

The strategic state is succeeded by the **visionary state**, where the organization reaches the highest order of reliability maturity and is able to drive broader efforts within and outside the company, based on their best practices and experiences.

Levels

Absent

Product functionality is the main driver for the Absent organization, with reliability being secondary.

The Absent organization may be steeped in processes and bureaucracy, but many or most of them do not incorporate reliability as a consideration. For example, the team's design review process or the PMO's gated project process does not ensure that reliability is a consideration of every facet of the desired product. These processes focus more on the features to be delivered, without considering that reliability can be baked-in to ensure a stable user experience. The organization does not have a formal process to write postmortems or retrospectives when things go wrong with the product, and may address reliability only after a very large outage. Even then, these are often addressed with only bandaid-type fixes, and the organization does not contemplate longer-term remediations that could benefit not only the product that went down so spectacularly but also other products which may have similar risks.

This type of organization may be able to describe the usage of a feature based on metrics, but cannot tell you the reliability of that feature.

Forward thinking in the organization does not contemplate risks to the service after it launches, but rather, is heavily involved in measuring the risks to roll out features.

The organization recognizes those who work on feature or product launches through bonuses, public callouts, and promotions but does not reward folks for technical debt reduction nor reliability improvements.

This reliability phase may be appropriate for products and projects that are still under development.

Reactive

The Reactive organization rarely has longer-term investments in fixing system issues. Responses to reliability issues/risks are tied to recent outages, with sporadic follow-through.

This organization may have some reliability metrics, but they are not comprehensive of all user experiences. The metrics do not lend themselves to deriving meta issues across the landscape. It is highly unlikely that the metrics have targets such as SLOs.

If there is an outage, a postmortem might be written, but there is no formalized process for reviewing postmortem quality nor tracking the action items that follow. The action items are left to the product owner to remediate and are easily dropped in the feature process. Post launch is when new alerts are added based on issues that arise, but they are not part of the launch criteria.

The organization does not spend time exploring for new or unseen risks. They only react when a risk becomes a reality. System complexity may be known, but it is likely considered too difficult to resolve. Complexity continues to grow unabated as a result. Design templates may have a reliability section required, but the contents are not considered significantly during design reviews. Reliability is understood on a per-system basis, but the landscape with upstream/downstream dependencies are not known.

Reliability work is not as valued as feature launches and may not be rewarded in the performance evaluation process. If reliability is recognized it is almost certainly related to heroes for work during outages.

Reliability is not a focus of leadership except during or just after a large outage. There is no regular cadence of reliability reviews by executives and it is not a core value of the team. The absence of outages is not celebrated and may be considered lucky.

This phase is appropriate for products/projects in a pre-launch or stable, long-term, maintenance phase.

Move from Absent to Reactive

The move from an Absent to a Reactive phase of reliability maturity is the one that is closest to a checklist. Based on the attributes from Absent, there needs to be a physical manifestation of reliability in an organizational structure and process.

The migration from Absent to Reactive can take months to a year.

- Staff a dedicated team on reliability, focused on the following (this may or may not be an official Google-style SRE team, but the team should be focused):
    - Define metrics for major services and set a target to improve reliability. This is typically in the form of an SLO/SLI, but can take other forms.
    - Gate all changes based on the impact to this reliability metric.
        - There is investment in stabilizing the metric and picking a non-noisy threshold to manage.
    - Monitoring and alerting for the metric: strongly suggest that the development team be responsible for responding to the alerts of this metric. They have the most influence on making changes that move this metric into compliance.
- Focus on getting an on-call rotation set up for production issues, and put escalation processes in place. If possible, do these using development team resources.
- Bring the postmortem mindset to the team: write, review, and aggregate postmortems, and reward well-written postmortems.
    - Have the reliability team and developers read Blameless Postmortems and a Just Culture.
- Do a quarterly postmortem, postmortem action items, and metrics review with leadership, to ensure that teams know that this work is important.

Case Study: New Product Launch: Embrace reliability principles from the start

A team with a new user-facing product was focused on adding features and growing their user base. Before they knew it, the product took off and saw exponential growth.

Unfortunately, their laser-focus on managing user requirements and growing user adoption led to high technical debt and reliability issues. Since the service didn't start off with reliability as a primary focus, it was very hard to incorporate it after the fact.

Much of the code had to be rewritten and re-architected to reach a sustainable state. The team's leaders incentivized attention to reliability throughout the organization, from product management to development and UX domains, constantly reminding the organization about the importance of reliability for the long-term success of the product. This new mindshift took years to set in. However, they were able to reach a reactive state with a focus on end-user journeys and active executive interest in postmortems and remedications.

This case shows phase migration as follows: Absent > Reactive

Proactive

The Proactive organization's potential reliability risks are identified and addressed through regular organizational processes.

A proactive organization has a comprehensive set of reliability metrics and standards for how code is written and run in production. Most of the production outages are reflected in these metrics, and there is a clear definition of the severity levels for an outage based on these metrics—this is usually in the guise of SLOs. Ideally, SLOs are based on Customer User Journeys (CUJs) and the new CUJs drive new SLO definition by process. During outages, escalation paths are well defined in the organization and in dependent organizations. Each outage has a postmortem that is led by leaders, and resulting action items are tracked and driven to resolution.

Given the organization has processes to manage outages, postmortems, risks and resulting action items, the impact to developer productivity is known and predictable. This time is tracked to address upward trends. Testing and release practices minimize the impact on developer velocity where possible. The organization knows that it takes developer time to ensure system reliability, but that impact is driven down proactively.

Potential reliability risks are identified and addressed as part of the regular program. Design docs are written with reliability in mind and are reviewed for known risks and failure modes to ensure that graceful degradation is built into the system as it changes. Single points of failure are not introduced by ongoing feature work, complexity is actively managed, and data flows are easy to understand and simple to debug.

Leadership continues to prioritize reliable efforts which not only include participation by all cross-functional teams such as production engineers, but also software, quality, and product managers and UX teams. Rarely does the organization need to spawn emergency task forces to address rising reliability concerns. Leadership regularly reviews monthly and quarterly reliability metrics, including postmortem quality and postmortem action item burn down.

Yearly and quarterly planning cycles reflect a concerted effort to inject reliability remediations based on data-driven analyses.

Leadership recognizes proactive reliability work and is comfortable rewarding downward trends or the absence of outages.

Most services/products should be at this phase, particularly if they have a large blast radius or are critical to the business.

Move from Reactive to Proactive

This move can likely take 2+ years to complete.

Initial Stages

- Focus on improving the quality of the reliability metric. Your metric should be impacted when there is an outage, and you can alert on this metric to indicate an issue.

- Continue the work on postmortems—write, review, and aggregate them. Ensure there is an automated tracking of action items so they get addressed in a timely fashion.
- Ensure that action items address the creation of automatic detection mechanisms, to minimize issues being found by end users.
- Include postmortem action items in the development workflow, to draw attention to them.
- Instead of a single team driving reliability reviews, hold quarterly reliability reviews that are presented by different component owners, to discuss the reliability of their systems. That way, you are gradually training all your leads so that they care about reliability.
- Publicize reliability wins across the org and reward or show recognition to people who worked on reliability. Brag about it!

Later Stages

- Document all your dependencies and know their SLOs.
- Engage with your dependent organizations to ensure they are meeting your requirements through quarterly business reviews, feature tracking, and so on.
- Do a risk assessment for all your components.
- Try to pick a couple of risk areas and staff efforts to address them.
- Understand your developer velocity and be able to measure how reliability work has an impact on that, such as slower rollouts, more testing rigor, etc.

Case Study: Mature Product: If you think you are done, think again

End users were highly dependent on the reliability of a key Google product, which tied directly to user trust. For this reason, reliability was top of mind for this Google organization for years, and the product was held as the gold standard of reliability by other Google teams. The org was deemed visionary in its reliability processes and work.

However, over the years, new products were added to the base service. The high level of reliability did not come as freely and easily as it did with the simpler product. Reliability was impacted at the cost of developer velocity, and the organization moved to a more reactive reliability mindset.

To turn the ship around, the organization's leaders had to be intentional about their reliability posture and overall practices. For example, they considered how much they thought about and prioritized reliability, and to what extent they were willing to slow down developer velocity to improve reliability in the short term. Even with vestiges of the prior reliability culture, because of the technical debt, significant number of new people, and the increased complexity of the system, it took several years to move the team back to a strategic mindset.

This case shows phase migration as follows: Visionary > Reactive > Proactive > Strategic

Strategic

Organizations at the Strategic phase manage classes of risk via systemic changes to architectures, products, and processes.

When an organization is strategic, they are looking at leading indicators of risks to reliability such as testing coverage, pathological queries, release and cherry-pick rates. Metrics start to reflect impacts that are smaller, mitigations are faster, and there are near-misses instead of outages. The organization has regular reliability cadences with dependent organizations to understand their roadmaps and reliability metrics to ensure that reliability positions don't degrade. This organization shares outage learnings and design changes across other products to share best practices.

Strategic organizations understand well the tradeoffs they are making between reliability, efficiency, latency, and developer velocity, and improvements to one area do not affect others significantly. All launches and production pushes are gated with reliability metrics, with slower rollouts that can be stopped if indicators are off.

The organization is studying classes of risks on a regular basis and addressing them holistically, not only for their own products but for all their dependencies. Design practices have foundational principles established for robust systems and analyze the dependency structure to reduce or improve dependencies. Logic reuse is a common practice and one-off, bespoke solutions are scrutinized to eliminate them. Experimentation support is built into the design, and features and risky code can be isolated with flags. Residual risks are regularly analyzed and addressed; the organization does not sit on the lower-level risks.

All teams have active discussions about reliability and feel accountable for maintaining it. Leadership is not the only voice driving these questions. All workflows, priorities and vocabulary is centered on reliability, which is a core value. Reliability is celebrated equally with features and launches.

This phase is appropriate for services and products that need very high availability to meet business-critical needs.

Move from Proactive to Strategic

The move from Proactive to Strategic can take two or more years to ensure that the culture is set and the architecture of the system can be adjusted.

Initial Stages

- Keep an eye on Mean Time To Resolution (MTTR), and the majority of the outages are detected by automated alerts.
    - Ensure that action items create automated mitigation logic such as automated rollbacks.
    - If your MTTR is varying widely, then new types of outages are likely being introduced regularly. Focus on future risks and proactively prevent them from becoming issues. If MTTR is frustrating to drive down, then focus on improving mitigation techniques such as automated rollbacks, canarying, feature flags, etc.
- Summarize meta trends across multiple postmortems for clues into where the ROI might be for larger initiatives.

- Start building a set of leading indicators that notifies you before reliability is affected. For example, SLOs are degrading but have not been violated by default (SLO slow burns).
- Every six months, do a risk assessment and make sure you are making progress on reliability efforts.
- Systems can gracefully degrade due to dependent outages; Single Points of Failure (SPOF) are eliminated.
- Continue talking about reliability metrics during org-wide forums.
- Do a reliability review at every launch.
- Be comfortable celebrating the absence of outages.

Later Stages

- Focus on understanding dependencies and reliability risks posed by your dependencies; fund remediation jointly.
- Staff people for longer-term architectural changes and really look at the following:
  - Architecture is designed with testing and debugging in mind
  - Graceful degradation and mitigation techniques built into the system
  - First-class experimentation support built into the design (including counterfactuals)
  - Ability to isolate features/risky code
- Evolve the risk assessment matrix and prioritize risks and remediation during all planning cycles. Allocate time on each team for this work.

Case Study: Nobody can be a hero forever

One infrastructure services team started small with a few well understood APIs. One key member of the team, a product architect—let's call them Sara—understood the system well and ensured that things ran smoothly by ensuring that the design decisions were sound. Sara was the one at each major incident to rapidly mitigate the issue, and it was they who understood the entire system and was able to predict what can and cannot impact its stability. However, when Sara left the team, the system complexity grew by leaps and bounds. Suddenly, there were many critical, user-facing, and internal outages and it was difficult for anyone to really understand the entire system any longer.

Organizational leaders initiated both short and long-term reliability programs to restore stability. They focused on reducing the blast radius and the impact of global outages. Leadership recognized that to sustain this trajectory, they had to go beyond engineering solutions and implement cultural changes, such as recognizing reliability as their number one feature. This led to broad training around reliability best practices, incorporating reliability in architectural/design reviews, and recognizing and rewarding reliability beyond hero moments.

As a result, the organization evolved from a reactive to a strategic reliability mindset, aided by setting reliability as their number one feature, recognizing and rewarding long-term reliability improvements, and adopting the systemic belief that reliability is everyone's responsibility—not just that of a few heroes.

This case shows phase migration as follows: Reactive -> Proactive -> Strategic

Visionary

The Visionary organization has reached the highest order of reliability and is able to drive broader reliability efforts within and outside the company (e.g., writing papers, sharing knowledge), based on their best practices and experiences.

Visionary organizations have systems that are self-healing and architectural improvements for reliability positively impact developer productivity as well (e.g., release velocity). This can be in the form of development frameworks that have reliability built in automatically.

Very few services or products are at this phase but when they are, they are industry-leading. Google Search was once considered industry leading and really set the tone for being the standard we all hold to. Now, there are several products at Google which are considered industry leading. Netflix with its reliability monkey was industry leading at one point, for another example.

Move from Strategic to Visionary

The key attribute of Strategic is that systems are self-healing; organizations should focus on that ability. In addition, improvements should positively impact the productivity of the organization. Both of these attributes result from the investment in service and development frameworks supporting the reliability goals of the organization. These framework investments likely benefit other products and services, given their flexibility and robustness, and can be shared inside the company, as well as externally.

Conclusion

It is common for an organization to have attributes from multiple levels of maturity. The organization may be somewhat proactive but have vestiges of Reactive or even Absent. The organization should focus on those areas to ensure that they continue to sustain Proactive. Getting developers and SREs involved is the most effective way to maintain the desired level. If the organization relies on checklists, processing, and program managers to drive reliability, it will not be sustained. Leadership has to be genuinely vested across all areas of the organization to maintain a higher level.

It's important that cross-functional organizations be honest about their reliability journeys and determine what is appropriate for their business and product. It is not uncommon for organizations to move from one phase to another, and then back again, as the product matures, stabilizes, and then is sunset for the next generation. Getting to a strategic level can be 4+ years in the making, and require very high levels of investment from all aspects of the business. Leaders should ensure that their product requires this level of continued investment before moving forward with it.

We encourage you to study your culture of reliability, assess what phase you are in, and determine where you want to be on the continuum. Then, carefully and thoughtfully move there. Changing culture is hard and cannot be done by edicts or penalties. Most of all, remember that this is a journey and the business is ever-evolving; you cannot just set reliability on the shelf and expect it to maintain itself in perpetuity without making time for it.

Works Cited

[1] Adkins, Heather, et al. *Building Secure and Reliable Systems*. O'Reilly Media, Inc., 2020 (ISBN: 9781492083122).

[2] Gellar, E.S. “Ten Principles for Achieving a Total Safety Culture.” *Professional Safety*, September 1994, pp. 18-24.