WHAT EVERY CEO NEEDS TO KNOW ABOUT

SLOs

NOBL9

February 2021 R12

This document, “What Every CEO Needs To Know About SLOs” is licensed under Creative Commons: Attribution - Noncommercial - Share Alike.

TABLE OF CONTENTS

Executive Summary

Site Reliability Engineering (SRE) is an operating model that helps your organization grow and innovate while maintaining infrastructure reliability for service levels that keep customers happy. Those service levels are defined by SLOs, or service level objectives, which set measurable bounds for what customers are willing to tolerate.

There's a lot that goes into establishing an SRE strategy both from the perspective of DevOps processes and an organization's business goals. This ebook will outline exactly what CEOs, CFOs, and others in an organization's leadership roles should know about SRE, and why they must play a part in mapping out the processes by which their teams will define and manage SLOs for their organization.

This ebook is designed for executive audiences, focusing on four critical areas that define how SLOs play an important role in enabling their organizations to deliver software that delights customers with measurable infrastructure cost management metrics.

Site Reliability Engineering is a Collaborative Effort

On the surface, the term "site reliability engineering" seems like it should be handled by a team of engineers. However, CEOs should be just as involved in the reliability process as any member of the devops team. This ebook demonstrates how SLOs define the important limits on what your customers are willing to tolerate before becoming discouraged quitting your brand altogether, and give IT operations teams the freedom to make data-informed decisions within those predefined bounds. Once your company adopts an SRE strategy and sets its SLOs, you can frame the outcomes of your monitoring data in terms a CEO can boast—customer delight, efficient revenue growth, smooth operations, and speedy product development.

Site Reliability Engineering is Smart Resource Engineering

Readers of this ebook will gain an understanding of how SRE provides a data-informed approach to delivering what customers want within the bounds of the imperfections they're willing to accept. Although 100% availability seems ideal for any service, SRE helps CEOs understand that requiring 100% availability at all times is not only difficult to achieve, but also extremely costly. By reducing reliability goals a bit below perfection, your company can still achieve excellent customer satisfaction while also maintaining costs—another benefit in the eyes of a CEO.

Your Organization Can Create an SLO Today

When you're ready, let SRE build a bridge between your IT operations teams and other business stakeholders, start with defining SLOs through user stories. After reading this ebook, CEOs and CFOs will understand how to phrase the customer experience issue more precisely as a measurable reliability goal, and plot the value at risk with the cost of reliability. The break-even point signals the business-justifiable SLO to sustain your customers' happiness. Because SLOs are created through collaboration by multiple teams in the organization, they will encompass considerations on a variety of priorities and tradeoffs based on data. This will set your business on a clear path towards keeping your customers happy.

Keep the Momentum Going and Define More SLOs

Don't stop after you've completed this ebook and created your first SLO. As you progress with your newly-established SRE strategy, you'll notice that adopting additional SLOs for other areas—product releases, for example—will have an impact on the culture of your organization. Quantifiable reliability metrics and risk management will become the norm, and will set a foundation for efficiently managing the growth and success of your business.

How To Optimize Gross Margins For Digital Services

Originally published on Forbes.com on December 3, 2020
Author: Kit Merker

Business leaders today need to reduce downtime in digital delivery to customers. According to IHS data, downtime costs North American businesses some $700 billion per year now that the world has gone digital. Microsoft CEO Satya Nadella famously said, "We've seen two years' worth of digital transformation in two months." Your customers expect digital experiences to "just work" and may leave you for a more reliable competitor. But is it also possible to have too much of a good thing? If you try to make every part of your system work correctly, you may quickly realize the laws of diminishing returns apply to you.

Too Good To Be Noticed

What if you overbuild your cloud infrastructure and engineering processes to create excess reliability, far above what any customers would even notice? Low reliability tends to get the attention, with news about massive outages and SLA violations making headlines. It's no wonder that businesses worldwide will be throwing an estimated $2.3 trillion by 2023 into digital transformation. A big part of this investment is likely to make the services work reliably for all customers.

There is a massive gap between excellence and perfection. And as any financially minded investor knows, without risk, there is less room for gains. By striving for perfection, you are ridding your company's services of downtime risk but killing the opportunity to get outsized margins.

The question is this: How do you lower cost to deliver while maintaining customer acquisition, conversion, retention and satisfaction?

Cost To Deliver Versus Cost Of Downtime

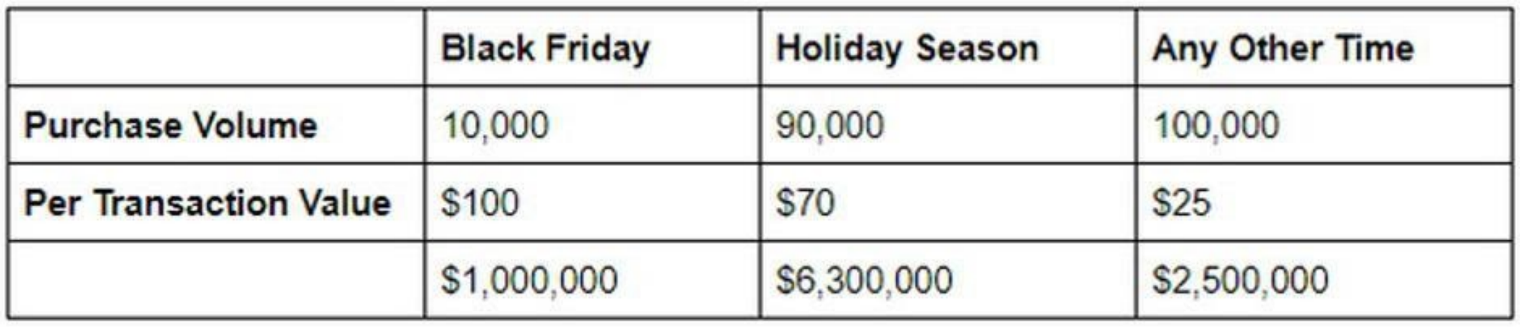
Your cost to deliver includes all costs directly related to hosting, serving and operating your digital services. This metric is important because it will tend to grow exponentially with your increases in reliability demands. Cost to deliver is different from cost of downtime, which is the revenue you would hypothetically lose during an outage. This metric can be quite misleading because depending on when the downtime occurs, which customer segment it affects and what transaction they were trying to accomplish, the cost of downtime can vary widely. On the other hand, the cost to deliver metric has an unmistakable pattern. For a given amount of load on the system, the cost to deliver will increase exponentially as the reliability target rises toward 100%.

Justifying Reliability Investments

Business leaders should not let fear rule their decisions to create growth and sustainability in business. Too often, we can't financially justify expensive investments in technical infrastructure. I suggest that you take the following steps when evaluating these projects.

First, break down the problem into specific groups of customers, geographies and demographics. Look across the various transactions (both monetary and nonmonetary) that customers are trying to accomplish on your website or app, and ask how likely they are to have a real business impact. Here is a hypothetical example we will use for this discussion from an online seller of shoes.

U.S. Shoe Shoppers Purchasing Via Website On Black Friday



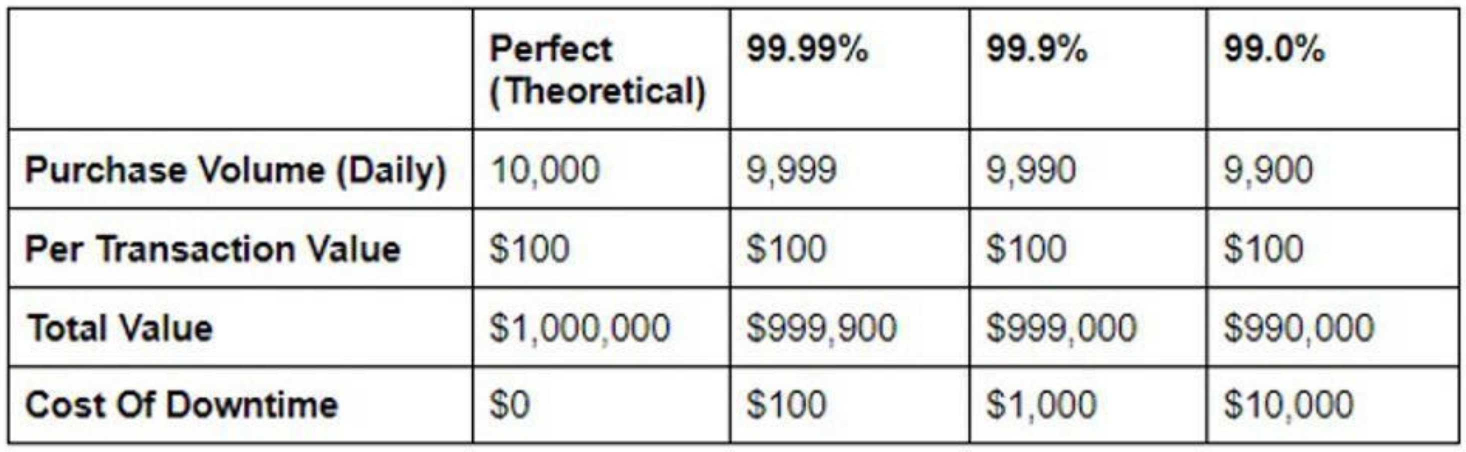
	Black Friday	Holiday Season	Any Other Time
Purchase Volume	10,000	90,000	100,000
Per Transaction Value	$100	$70	$25
	$1,000,000	$6,300,000	$2,500,000

Table 1: U.S. Shoe Shoppers Purchasing Via Website: *Image courtesy of Kit Merker*

Use this breakdown to analyze the risks that would be resolved by the requested budget (both people and cloud/external SaaS services) to understand the specific ROI. Your team may be trying to justify the expense using an average cost of downtime, which won't hold up to scrutiny in this new breakdown. You must understand how the cost to deliver will change after the project and compare it to the current and expected revenue and reputation improvements. Let's look just at Black Friday in the table above and see what would happen at different reliability levels for these purchases.

Please note that these are worst-case scenarios where the transactions were not retried. Often a customer will simply retry a failed transaction (but not repeatedly).

U.S. Shoe Shoppers Purchasing On Black Friday: Sales Risk Analysis



	Perfect (Theoretical)	99.99%	99.9%	99.0%
Purchase Volume (Daily)	10,000	9,999	9,990	9,900
Per Transaction Value	$100	$100	$100	$100
Total Value	$1,000,000	$999,900	$999,000	$990,000
Cost Of Downtime	$0	$100	$1,000	$10,000

Table 2: Sales Risk Analysis Of U.S. Shoe Shoppers Purchasing On Black Friday: Image courtesy of Kit Merker

Now that we understand the risks of this transaction not being completed at various reliability levels, we can understand the ROI of a proposed infrastructure budget. Let's say

your engineering team is scoping a $50,000 project that will improve reliability from 99.0% to 99.9% in this transaction. You will quickly see that it will take about five Black Fridays to pay for this project. Perhaps the investment would apply year-round, in which case the $63,000 lost from the holiday season and the $25,000 lost from the rest of the year (see table one) would easily justify the infrastructure investment. But if this same $50,000 project were to go from 99.9% to 99.99%, you would not even be close to breaking even.

Summary

To achieve sustainable growth and maximize gross margins, you need to be smart about allocating resources to reliability projects. Counterintuitively, by reducing reliability goals a bit below perfection, you can achieve excellent customer satisfaction while maintaining costs.

I advocate for excellence in digital services, and I believe businesses must meet customers' basic expectations. However, once you achieve this, there is a limit to the value of additional reliability. Prioritize investment projects around specific hot spots where there is a high risk of downtime having a business impact and where and the cost to deliver is less than the risk.

Three Steps For Measuring Customer Experience In A Digital Business

Originally published on Forbes.com on April 21, 2020
Author: Kit Merker

No matter what industry you're in, your business is becoming increasingly digital. Managing your digital service requires key performance indicators (KPIs) that ensure you provide a great experience to your customers.

KPIs are used in business all the time to quantify and track the health of the business. Traditionally, you might review these indicators on a daily or weekly basis and then aggregate them to a quarterly or annual view so you can understand longer-term trends. Today, you need to measure online digital services that you deliver through software in the cloud or on mobile devices using a much shorter timescale — sometimes even milliseconds.

Every business, from movies to retail and financial services to food delivery — or even manufacturing — has software-driven experiences that need to work to keep the business running. Since we talk about these businesses delivering "whatever as a service," it makes sense to define your service KPIs. Rather than just collecting lots of data (which is tempting with software services), you should instead define a small set of service KPIs that tell you a lot about your customer experience in terms of the reliability of your service.

Another important aspect of delivering a service is making sure your technology teams know the business expectations that will keep customers happy. For example, it might have been OK a few years ago to announce planned downtime every week for a couple of hours while your teams install the new version. Not so today: If you want to run an "always-on" service that's as good as your competitors, you will likely need to push the organization to design and deliver an installation system that requires no downtime.

In a high-reliability expectation environment, Service KPIs usually have targets somewhere between 99% and as close to 100% as possible. However, setting a reliability target at 100% is not only unrealistic, but it also often doesn't make business sense. Setting a target too high means you could see an exponential increase in costs and effort to deliver at this level, and your customers are unlikely to notice anyway. In my experience, most internet connections to your service will run at a mere 99.9% reliability, which means you can expect about 43.3 minutes per 43,800-minute month of downtime or slowness. By contrast, 99.99% (four nines) means only 4.3 minutes in the same period.

How can you define these Service KPIs? As the SVP of business development at a company that provides reliability and service-level management technology, here are three simple steps I recommend.

1. Figure Out What Matters Most For Your Customers

Ask yourself: "What absolutely must work to keep customers happy?" Look across your service and think about the sign-up experience, the login experience, and the ever-frustrating password reset experience. If you are delivering video, you might need to look at the movie streaming quality. If you are delivering food, it might be the ordering experience.

This element is unique to your business and reflects what matters to your particular customers. Try to put yourself into the mindset of a customer and what they are experiencing. What is going to make them upset if it doesn't work?

Once you have your top three to five essential user experiences, take a look at how these service KPIs correlate to real-world indicators of user satisfaction. Find periods of time that included known downtime, when these Service KPIs were definitely low, and compare them to various pieces of historical data, including customer support tickets, social media sentiment analysis or even purchases and cancellations. A great set of KPIs will be lower when these external metrics spike. The most common starting points are measuring if the service is "up" (called "availability" by tech folks) and if the system is "fast" (sometimes referred to as "latency"). But don't let this limit your ideas for service KPIs.

2. Define A Measurement Strategy For Each

Once you've defined the service KPIs, you will need to work with your technology team to collect and report this data in a central place. You may be tempted to include your support tickets and Twitter sentiment data in your service KPIs. While it's a good idea to review this data regularly, you'll want to keep it separate. That's because it is a lagging indicator (the service must be broken before people complain), but it will serve as a valuable calibration tool for your service KPIs over time. If you learn it will be difficult or impossible to measure the service KPIs directly, then refine them into a "proxy metric" that is close to your desired service KPI. Defining these metrics is a bit of art and science.

3. Implement Measurement And Cross-Functional Reviews

You can use this data to make important decisions about your feature road map and help your team to correctly prioritize reliability improvements when the service KPIs are not meeting expectations. As you are now measuring the quality of the experience of a broad set of customers, and understanding their experience in terms of the reliability of the digital service you are able to provide, it's important that this data is accurate and that you use it in data-driven discussions about how to optimize your service.

Adapt To Digital Service Expectations

As your digital service business grows, you should work to keep adapting by delivering a great service at increased demand as efficiently and with as few hiccups as possible. Being able to reliably deliver that service is arguably the most important feature of your service and will encourage your customers to keep coming back for more. If you can't define and measure the customer experience and get everyone on board with the idea that these metrics matter, you will be driving blind. Take these simple steps to get started, and then commit to reviewing and refining your service KPIs regularly so you can protect your most valuable asset — your customers.

SRE 101: The SRE Toolset

Originally published on Nobl9.com on May 13, 2020
Author: Kit Merker

What tools do you need to get started with SRE?

Much has been written, especially by the founders of the Site Reliability Engineering (SRE) concept at Google, about the benefits of all parties in an organization working together to balance features and reliability. We recognize that there are two opposing forces that each make perfectly good sense: on the one hand, your company (especially your application developers and software engineers) wants to maximize the speed and agility of deploying new applications and product features. On the other hand, your company (especially your IT operators) also wants to maximize the reliability and availability of its infrastructure and minimize downtime that could adversely affect customer satisfaction.

Unfortunately, in far too many organizations, these two equally valid objectives are being pursued by siloed departments at cross purposes—and that operating model produces substandard results, frustrated employees, exasperated senior execs, and, worst of all, unhappy customers.

It doesn't have to be that way. SRE builds a bridge between product managers/application developers and IT operations. SRE is a practical, data-centered way to find common ground between Dev and Ops, so that both functions can align on objectives that achieve optimal customer satisfaction.

To learn more about how SRE came to be, I'll refer you to the source, but here, I'd like to get down to brass tacks: identifying the three most essential tools in the SRE toolset—observability/monitoring, incident response, and service level objectives (SLOs).

1. Observability/Monitoring

Even if your role is not IT/ops, you probably understand the concept of monitoring a system to gain insight. Monitoring is about collecting metrics from a production system to gain an understanding of what's really going on. In cloud operations, monitoring is particularly helpful for debugging and triggering alerts when something needs attention. The challenge of monitoring is separating the true signal (the few, critical things that actually need attention) from the noise (the many false signals that are at best a distraction and at worst another reason to hate pager duty). Of course, all this becomes even more difficult as your system scales.

The concept of observability is similar to but slightly different from monitoring. Observability is a measure of how well we can understand the internal state of a system by solely looking at its outputs. In other words, it is how well we can deduce internal causes by observing external symptoms. The more observable our infrastructure is, the more success we will have in diagnosing and curing problems that arise. One way to improve the observability of a system by providing more context in the log outputs, giving our monitoring systems greater insight into what's really going on.

2. Incident Response

One of the core tenets of SRE is making systems that are automatic, not just automated. That is, we aim for systems that run and repair themselves, and we want to intentionally minimize human involvement in the system so that operations can scale. Therefore, it may seem odd that incident response (by humans) is a primary tool in the SRE toolkit.

Here's why: Failures are inevitable*. You can automate responses to anticipated failures, but there will always be new causes for a new type of failure you didn't anticipate—a new software release, a spike in demand, an outage in a third-party system. When you're dealing with infrastructure, Murphy's Law applies: Anything that can go wrong will. And Finagle's Corollary applies as well: Anything that can go wrong will, *and* at the worst possible moment. The point of having a good incident response tool in your toolkit is to minimize incidents, prevent staff burnout, respond to incidents as efficiently and effectively as possible, and conduct post-mortem analysis to enhance staff training and fuel continuous improvement efforts.

3. Service Level Objectives (SLOs)

You've probably been tracking right along with me so far in this discussion ("Monitoring, yep. Incident response, got it."). So, don't let me lose you here, because this is by far the most important tool of the three.

In fact, SLOs are Job one for SREs.

Let me ask you this: does your IT operations team have an availability goal? Is it realistic? Is it too high or too low? Is it intrinsically aligned with the desires of customers? Are you confident that your IT infrastructure can scale to meet the needs of your customers and the needs of your business? Most importantly, how did you set that availability SLO? What process did you use to get there?

These are all great questions. And, if you're like most people in your shoes, you've not thought much about them. Or, if you have, it's highly unlikely you've incorporated a formal process into cultivating answers to these questions that feed into your process.

Keeping infrastructure available (reliable) is an essential task—without it, no one can really trust the infrastructure in the first place! Yet, in many organizations today, we can't seem to get ahead of inevitable reliability issues, and the only way we are learning to improve the system is when it breaks.

Well-constructed SLOs are the tools you need to solve this problem. SLOs are strong, understandable promises made to customers that their applications will run reliably and with adequate performance on a cloud. What makes SLOs uniquely valuable and powerful is that they are created by a collaborative team of application developers, product managers, IT operators, and other stakeholders in a business by strategically considering priorities and tradeoffs.

What do I mean by priorities and tradeoffs? Here is a non-IT example that illustrates the concept: Suppose you need to rent a room for a party. How large does the room need to be, and how much of your party budget can you allocate to room rental? If you opt for a smaller, cheaper room, what are the downsides if it turns out to be too small to hold all of your guests?

In this (admittedly abstract) example, we must make a judgment about the tradeoff in capacity versus cost. In the world of cloud computing and software, we have capacity versus cost tradeoffs too. We also have tradeoffs between speed of releasing new features versus avoiding errors/failures in the system. We are constantly making judgment decisions about tradeoffs, and too often we're making those decisions based on gut instinct, without the input of all the company stakeholders, and without understanding how it will impact the customer. SLOs, in stark contrast, help us make tradeoff decisions based on data, with the input of all the company stakeholders and centered on what it takes to delight the customer.

The SLO tool is the key to the ultimate value proposition for any business: products and services that keep customers happy.

In fact, SLOs are so important that we've built Nobl9 on that premise. Our business is about helping our customers precisely quantify the experience of the user, then statistically and rationally translate that knowledge into wise tradeoffs and informed resource allocation decisions. Or, in more practical terms, we're about:

- saving IT operators time and hassle
- helping developers accelerate time to market
- reducing spend on unimportant infrastructure initiatives
- redirecting IT investment to initiatives with the highest ROI
- ...all while achieving a level of performance that delights the customer

  If that sounds like something that solves a problem in your organization, there are more useful resources here in the Nobl9 blog. Product managers might be interested in this post, and anyone helping teach the C-suite about SRE might want to read this one.
  * Read more about how Netflix adopted a new mindset of software failure as the rule, not the exception, and survived the big 2015 AWS outage.

How The CEO Should Think About SRE

Originally published on Nobl9.com on May 13, 2020
Author: Marcin Kurc

It's 2020, and your customers expect a lot of you: they expect the most innovative services, up-to-date websites, easy to use software applications, and intuitive mobile interfaces. And, they expect all of those things to be available and running efficiently upon their demand. If customers don't get what they want when they want it, they take their business elsewhere.

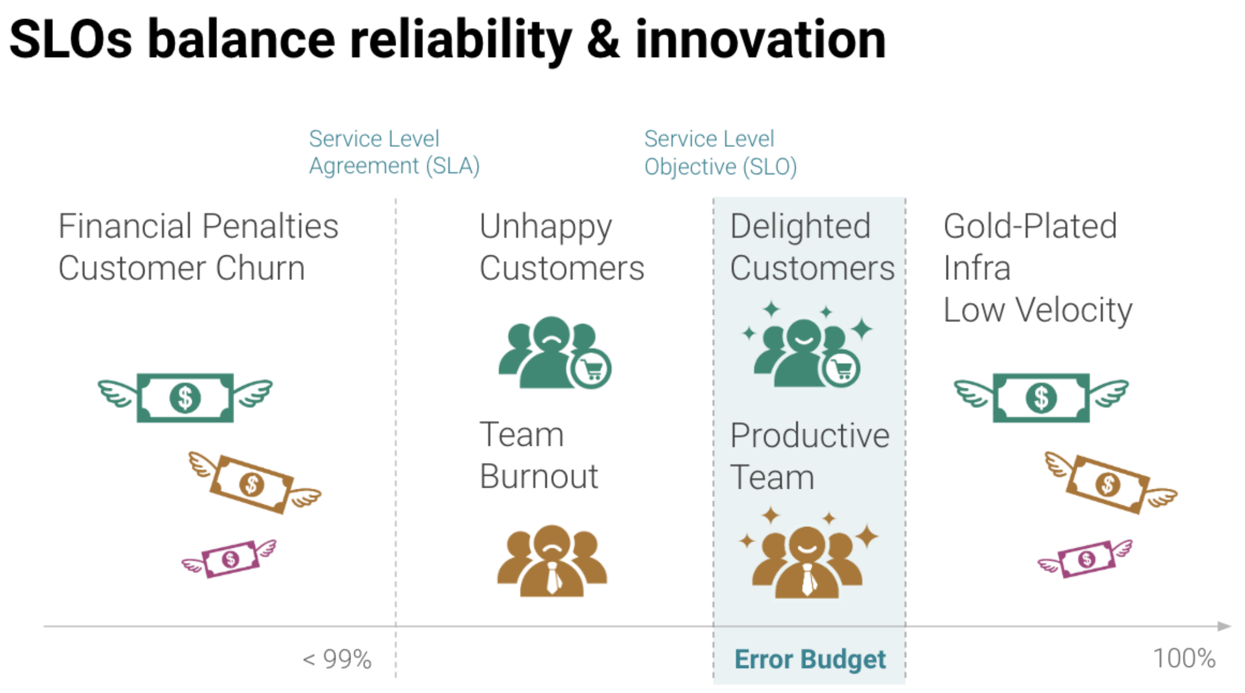
So, as CEO, you have a triple challenge: Delight your customers with high-quality services and rapid iteration of innovative features, *while* maintaining reliable delivery that meets customer expectations, *and* controlling costs to maximize ROI and the bottom line.

Piece of cake? Of course not! But SRE can help. Site Reliability Engineering (SRE) is a concept pioneered by Google (adopted by others and applicable to everyone) that is transforming the way product managers, business line managers, software engineers, and IT operators work together to please the customer.

In your organization, you may have noticed that the technical teams find themselves working at cross purposes with the business: On the one hand, product managers and business line managers want to maximize the speed and agility in getting new applications and product features to market; on the other hand, your technology teams need to maximize the reliability and quality of the infrastructure needed to provide those features consistently and safely to the customer. Both have valid priorities, and often it falls on you to make the call of where to invest.

Fortunately, SRE offers a way to institutionalize the best each these perspectives have to offer and develop an operating model and culture where all "sides of the house" collaborate to make data-driven decisions that are centered on customer delight.

What would your organization look like after adopting the SRE approach? Here's a glimpse of the future state once SRE is implemented, scaled, and operating smoothly:



Customer delight is king. Everyone in your organization understands that retaining customers is just as valuable (if not more so) than acquiring new customers and that it takes both to succeed. Now your software engineers and IT operators are working together, sharing responsibility and ownership of attaining customer satisfaction objectives.

Customer satisfaction is studied and articulated in measurable terms. Responsible stakeholders from across the organization are involved in the process of setting service level objectives (SLOs) that align with corporate business objectives. These data-centered SLOs dictate appropriate decisions and allocation of resources, especially when tradeoffs and compromise between departments are necessary.

Decision-making is powered by real data that really matters. Decision-makers are given less but more relevant data that truly correlates to business processes and business impact. No more guessing is needed because the data provided has a high signal to noise ratio. As a result, decisions are being made quickly and efficiently without unnecessary friction.

IT resources are being allocated wisely, with a full understanding of the risk/reward tradeoff. Investments are designed to make a service reliable enough, but no more reliable than it needs to be. SRE has helped you reduce spend on unimportant infrastructure initiatives, and redirect those resources to adding features, cleaning up technical debt, and reducing operational costs.

Smooth operations are the norm, so employee and customer satisfaction is at an all-time high. Automation is being used to automatically eliminate toil, hassle, and wasted time. Most potential problems are identified and resolved before they arise, which has greatly improved the working environment, especially for IT operators.

Your product development and DevOps teams are running faster, innovating more than ever before. SRE is helping developers accelerate time to market, delight customers, and improve your competitive position.

Bottom line: SRE is a no-brainer—a practical, powerful, and culturally palatable way to maximize customer satisfaction and revenue.

Now that you understand why a CEO should care about SRE, let's discuss dollars and sense. See "How to Explain SRE to Your CEO."

An Easy Way to Explain SLOs And SLAs to Business Executives

Originally published on Nobl9.com on May 13, 2020
Author: Kit Merker

If you appreciate the irony of TLA (a recursive acronym for “three-letter acronym”) this blog is for you. Even if you find the “alphabet soup” of business unappetizing, read on anyway, because we guarantee you’ll find some meaty morsels of value in the following discussion.

Nearly everyone, whether you’re in the C-suite or on the front-line of an organization, knows about performance metrics. They allow individuals, teams, and organizations in a business to focus on achieving goals that are important to the organization’s mission. Many types of performance measures are commonly used, and we typically refer to them with TLAs. Perhaps you’ve heard of these:

- KPIs—Key Performance Indicators: quantifiable measures used to evaluate the success of an organization, employee, etc. in meeting objectives for performance.
- SLAs—Service Level Agreements: contracts defining the level of service a customer expects from a vendor. These agreements lay out the metrics by which the service is measured as well as the remedies or penalties incurred should the agreed-on service levels not be achieved.
- OKRs—Objectives & Key Results: a simple management framework that helps everyone in the organization see progress toward common goals. Short, inspirational objectives define where you want to go. (Companies typically create three to five high-level, ambitious objectives per quarter.) Key Results are the deliverables that you define for each objective so that you can measure your progress toward achieving that goal. (Each objective should have two to five measurable key results.)

Recently, Site Reliability Engineering (SRE)—the infrastructure and operations discipline popularized by Google—has introduced several new TLAs to our business conversations, including SLO and SLI. Fortunately, we can easily understand SLOs and SLIs by drawing some comparisons to the business performance measures we already know:

The SRE term…	…is analogous to the business performance term…
SLO (Service Level Objective)	SLA. (An SLO is an SLA without contractual penalties.)
SLI (Service Level Indicator)	KPI. (An SLI is essentially a “service KPI.”)

An SLO is like an SLA

You can think of an SLO as an SLA without contractual consequences. SLOs are usually more stringent than SLAs because SLAs don’t necessarily map well to customer expectations. An SLO, by contrast, expresses the service level of infrastructure/operations we need to achieve in order to keep our customers satisfied. Commonly, SLOs pertain to infrastructure service attributes such as availability, latency, data freshness, or degradation. For example, an SLO for availability might address whether a website or application is available to be seen/used when the end-user wants to see or use it (e.g., “Customers using the application will find it available 99.95% of the time over a 1 month period”). The key to establishing an SLO is to “define the lowest level of reliability that you can get away with, and state that as your Service Level Objective.” To set the SLO any higher would be to waste money and time on providing an unnecessary degree of reliability.

Keep in mind that SLOs don’t carry any contractual consequences. These are penalties (usually financial) that must be paid to the customers if benchmarks are not met. SLAs do carry these penalties. Therefore, your SLAs should be less stringent than your SLOs (your internal service objectives). For example, if your SLO is an availability of 99.95% over 1 month, your SLA might be 99.9% over 1 month. That way, as long as you are meeting that internal SLO, you will, by definition, meet your SLAs and avoid penalties.

An SLI is like a KPI

When we measure the performance of a business, there are dozens of metrics we could use, but we typically focus on a few performance indicators that tell us at a glance how the company or a unit within the company is doing. These few metrics are selected because they best express what is truly essential to the company’s success. We call them *Key Performance Indicators*.

KPIs vary from one company and unit to the next, but some of the commonly used KPIs include profit, cost of goods sold (COGS), sales by region, annual recurring revenue (ARR), customer acquisition cost (CAC), employee turnover rate (ETR), net promoter score (NPS), and the granddaddy of them all EBITDA—earnings before interest, tax, depreciation, and amortization. Simple examples, but you get the point. Similarly, when we’re measuring the performance of our infrastructure, we want to focus on a few key indicators (service level indicators, or SLIs) that tell us the most about the user experience and where we are going to draw the line when we need to make tradeoffs between operational improvement and pushing new features.

Another similarity between KPIs and SLIs is that both are helpful in aggregating key bits of information. Just as the KPI “profitability” takes into account revenues and expenses, SLIs can tell you about multiple subsystems through a single metric. In this particular example, the indicator is a ratio. Ratios can be helpful in showing us the balance between conflicting pressures on the business. That tension between two priorities: expenses that drive customer satisfaction and revenue that’s needed to run the business. Another way to think about SLIs is that they are *KPIs focused on infrastructure services*, or “*service KPIs*.”

“What’s in a name? That which we call a rose | By any other name would smell as sweet”

If you’re sick of TLAs by now, that’s understandable. Even at Nobl9, our teams prefer to say simply “objectives” and “indicators,” and that works well for us. The names you choose does not really matter. What matters is applying the substance “SLOs” and “SLIs” does; it’s critical if you want to create highly reliable and scalable systems that simultaneously support the rapid development and launch of innovative services.

If you want more reliability content to share with the business stakeholders in your organization check out How the CEO Should Think About SRE and Dollars and Sense of SRE.

How To Explain SRE To Your CEO

Originally published on Nobl9.com on May 28, 2020
Author: Marcin Kurc

As we discussed in the previous post, Site Reliability Engineering (SRE) is an operating model that helps your organization grow and innovate with velocity while maintaining infrastructure reliability for service levels that keep customers happy. (It's like DevOps on steroids.) The benefits of SRE are many. The end result is customer satisfaction, and the bottom line is efficient revenue growth. Conceptually, it's a no-brainer.

But in order to champion SRE, a CEO also needs to understand the decision from a dollars-and-cents viewpoint and be able to defend it with all the stakeholders in the organization.

- How much is adopting SRE going to cost?
- How do you put metrics on the benefits you'll receive?
- How do you calculate the ROI?

At its very core, SRE is a framework for thinking about ROI and risk, so you might say that the dollars-and-cents analysis you need is built right in, in the form of SLOs and error budgets.

In SRE, service level objectives (SLOs) are defined for many different aspects of IT service to mark the precise level of service that needs to be achieved in order to avoid unacceptable levels of risk of displeasing the customer. An error budget, then, is the inverse of the SLO; it is the error rate we will tolerate for a given set of services, because we expect that error rate will not upset customers enough to warrant prevention. (The beauty of SRE is that these service-level objectives* are not arbitrary—they are tied directly to business outcomes.)

Let's use availability as an example. When we talk about how often your infrastructure is available (uptime), we typically speak in terms of "nines." If your infrastructure is available "four nines" or 99.99% available, it will be unavailable 52.6 minutes a year. However, if your infrastructure achieves "five nines," then your system is up and working 99.999% of the time—that is, it's down only 5.26 minutes a year. Take note that once you have multiple overlapping services and redundant regions, you need to calculate uptime differently as the proportion of customers served successfully. Measuring uptime in minutes may be overstating your actual reliability.

Avoiding Gold-Plated Infrastructure

In an ideal world, we'd want our infrastructure to achieve as many nines as possible; however, moving from one class of nines to the next higher class is roughly ten times more expensive (you'll incur significant people and infrastructure costs to make the leap to the next level). And, when you consider the inherent limitations of physics and the architecture of public networks, approaching five nines of reliability consistently can actually become very nearly impossible.

So how many nines are good enough? At what point on the "nines class scale" do your customers become unhappy with their service, that is, at what point do they notice and complain, or even walk away? In this case, the SLO is the uptime goal, and the error budget is a small acceptable allowance for the system being down.

SLOs and error budgets keep your customers happy while balancing the competing interests of product stakeholders who want to rapidly launch new features/products and IT operators who want to maximize infrastructure uptime. Here are two examples:

- In the case of a new deployment, both product teams and operators have to ask, if this deployment causes an outage, will we be within our error budget? If yes, then developers have the leverage to deploy; if no, IT operators have the leverage to say no. SRE gives you a green light to build features when reliability is under control.

- What if IT operators want to invest in upgrades for the sake of improved reliability? The question must be asked, "Is it worth eroding our margins for the sake of better reliability?" Or, put another way, "Am I throwing money at an issue that isn't a real problem?" (Am I paying for gold-plated infrastructure?)

You might be concerned about putting in an error budget system that you can't overrule. Think about error budget as a fiat currency, and you (and upper management) are the central bank. You can always "print" more error budget, but do it too much and you will devalue the currency!

So, in terms of dollars and cents, here's your cost/benefit equation:

- How much is it worth to you to retain a customer? Conversely, how much does it cost when you lose a customer? What value do you place on reducing customer churn? That's one side of the ROI equation you must consider when adopting SRE.
- The other side of the equation is the cost to adopt the SRE approach, and, to be fair, there will be operational costs incurred, primarily in terms of staffing and training. A good training option is the SLO Boot Camp.

Like everything else your exec team evaluates for organization-wide implementation, approach SRE from a cost/benefit perspective, and consider your risk profile. Properly implemented and operated, SRE frees your application teams to focus on delivering accelerated value to your customers. And, they can do this with new features and capabilities, within the risk-adjusted guardrails of SLOs that define what customers are willing to tolerate before bolting. At the same time, it gives your IT operations teams the freedom to make decisions about infrastructure management, unencumbered by the unachievable "never suffer an outage" standard that accomplishes nothing more than lining the pockets of your service providers and frustrating the best talents of your product managers and application developers.

In short, think of SRE as Smart Resource Engineering: a data-informed approach to delivering what customers want, within the bounds of the imperfections they're willing to accept. It's an approach that makes dollars…and sense! *SLOs are so important that we've built Nobl9 on that premise. Our business is about helping our customers to precisely quantify the experience of the user, then statistically and rationally translate that knowledge into wise tradeoffs and informed resource allocation decisions. If that sounds like something that solves a problem in your organization, there are more useful resources in the Nobl9 blog.

Monitoring Tells Me Nothing Says Your CEO

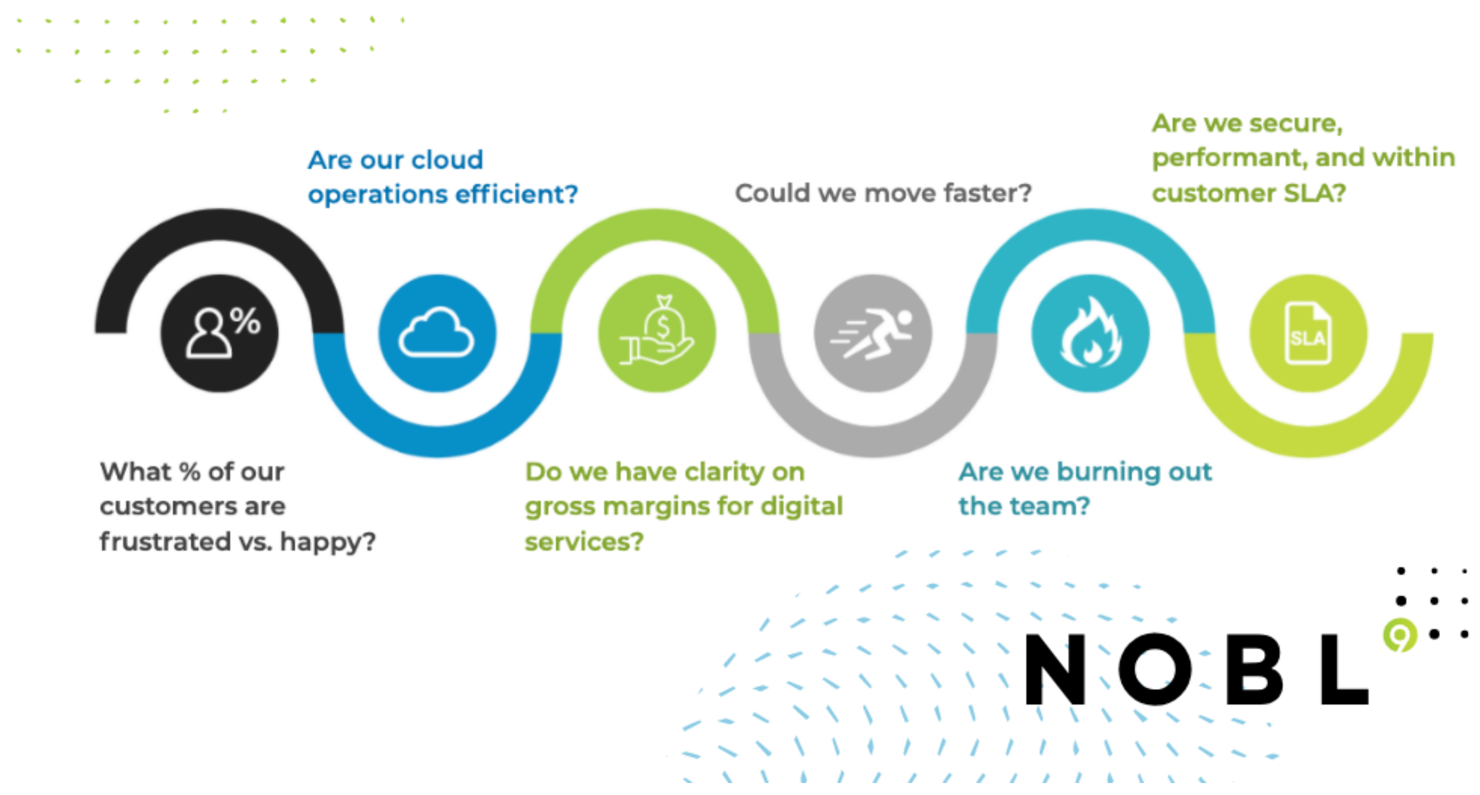
Originally published on Nobl9.com on July 6, 2020
Author: Marcin Kurc

Do you need to justify your infrastructure spend to your CEO? In this article, we'll equip you to communicate clearly and in CEO language about how the infrastructure investment you seek will benefit the business.

First, let's give your CEO some credit. She already understands that infrastructure has limited capacity. She knows if you get too many customers on your platform, it will slow down, negatively impacting your customers who are trying to log in and use the software. Nevertheless, saying to her, "we've got more customers, so we need more infrastructure" is a poor way to communicate the need.

And let's give you some credit too! Chances are, you are already measuring and monitoring your infrastructure, gathering data to identify specific infrastructure needs and help you prove your case for funding. However, don't be surprised if your CEO takes one look at your measurements and says, "Monitoring tells me nothing."

The problem here is that your CEO speaks a different language.


Are our cloud operations efficient?
Could we move faster?
Are we secure, performant, and within customer SLA?
SLA
What % of our customers are frustrated vs. happy?
Do we have clarity on gross margins for digital services?
Are we burning out the team?
NOBL9

What matters to your CEO are the questions depicted in Figure 1. Are we keeping customers happy? Are our cloud operations efficient? Are our digital services driving revenue growth? Frame your monitoring data in these terms.

If you are asking for $5 million more to spend on the cloud infrastructure line item, you must be able to express how that investment translates into reduced customer churn and positive marginal revenue.

Unfortunately, monitoring tools aren't designed to speak CEO. Just looking at dashboards provided by a Vice President of Infrastructure doesn't tell the CEO anything about the relationship of the request to business outcomes. Monitoring dashboards don't show the impact on margins, costs of running the business, customer satisfaction, etc. How does a CEO know if the infrastructure team is delivering value or just spending money?

What you need is a "translation machine" programmed with business logic to convert your infrastructure monitoring data into business terms that your CEO can understand.

Translating monitoring into business outcomes.

Such a translation device exists today. It's called SRE—software reliability engineering.

The principle concept behind SRE, as introduced by Google, is that the objective of the infrastructure team (and arguably the company as a whole) is to deliver the service the customer is expecting at the lowest cost. The main idea is to preserve "customer happiness," and that is something that CEOs understand—the value of keeping a customer happy (or, conversely, the cost of losing one) is bottom-line quantifiable. However, SRE also takes a very practical approach that delivering 100% perfect service is not worth the cost. In many cases, slightly less than perfect service is just as acceptable to customers and can be delivered at a much lower cost. Service Level Objectives (SLOs) are used to achieve the optimal balance.

SLOs are analytical tools used to define in measurable terms what is important to the customer and the level of tolerance customers have with service variations. SLOs can be applied to each of the components that comprise "customer happiness."

For example, an e-commerce application customer cares about whether they can log in instantly, skim good-quality product images quickly, and rapidly select items and complete purchases without glitches. Similarly, a gamer cares about real-time streaming and high-quality video. Each of these service components—latency, responsiveness, video quality, etc.—can be defined by SLOs.

Once customer-centric SLOs are in place for critical indicators of availability, latency, and overall quality, SLOs become the business logic processor we need to translate the performance of infrastructure into business outcomes. For example, when a particular service has elevated errors that put an SLO at risk (i.e., approaches a critical threshold), business executives and board members can see the problem in context, understanding that customers will be lost if the issue is not addressed. In a sense, SLOs formalize the "squinting at metrics" exercise and elevate the discussion above technical jargon to business outcomes.

If Only We Were a SaaS Startup

SLOs essentially let every company look at infrastructure the way a Software-as-a-Service startup does. When you are building a SaaS startup, you have a base set of infrastructure that has to scale, and as you scale, the cost of delivering the software goes up. When that's the core of your business, you can simply and directly correlate infrastructure costs to bottom-line business measurements (Churn, ARR, COGS, EBITDA, etc.).

In “real life,” when your non-SaaS business is far beyond startup stage, things get a lot more complicated and the connection between infrastructure spend and business outcomes becomes murkier. Too many times, CEOs are faced with making subjective decisions, relying on what people are telling them, rather than making decisions based on unbiased metrics truly tied to customer experience.

You can change that. The next time you take a spend request to your CEO, speak her language. Let her know how SLOs affect the cost projections and gross margins on the delivery of new, revenue-generating services. This time, she’ll say, “Now you’re talking.”

Why do I need SLOs if I already have monitoring?

Originally published on ToolBox.com on January 19, 2021
Author: Brian Singer

Let me start by saying I love monitoring. Monitoring is great. Monitoring—which is defined as collecting metrics to improve understanding of how a system behaves— is a significant source of the data we need to practice the customer-centric, SLO-based approach to software reliability. In that sense, monitoring and SLOs are complementary—they go hand in hand.

Monitoring helps us gather data to get a clearer picture of what's happening in our software and systems. If you've already invested in good monitoring tools like Datadog, Prometheus, New Relic, or others, that's great. Keep them. You need them. But you need SLOs too, and here's why.

Can Monitoring Tell You If You're Meeting Customer Expectations Over Time?

You can be using monitoring and alerts to inform you about the status of a system component at a moment in time, but that information by itself doesn't tell you whether the system is performing well or poorly from your customer's perspective.

Let's look at a simplified example:

How do you determine reliability over time? How about counting incidents, like pager duty alerts? If you have 20 incidents in Q1 and 10 in Q2, can you definitively say that you had higher reliability in Q2? Maybe. Or maybe not.

What if the incidents in Q1 added up to 120 minutes of unreliability but Q2 incidents added up to 150 minutes? In that case, your users would have experienced better reliability in Q1.

What if you took a Mean-Time-to-Resolution (MTTR) approach? Using the same data as before, we can compute that incidents in Q1 averaged 6 minutes each, and incidents in Q2 averaged 15 minutes each. So, does this mean that Q2 was more than 2 times as unreliable as Q1? That's also not true.

That's what's great about SLOs and error budgets: they give you the best way to report on reliability over any window of time, based on the real issues that customers actually care about. Or, put another way, SLOs filter and translate all that monitoring data and help you *identify precisely where to expend your energy to achieve the biggest return for the business*.

SLOs Guide More Advanced Efforts to Improve Reliability and Efficiency

Once you have your SLOs in place and have an error budget to spare, you can justify experimentation and low-risk chaos engineering (e.g., running tests to identify hidden dependencies in systems). SLOs help you determine when to schedule load and stress tests and even when to try turning systems down or off. With SLOs, you can be proactive rather than reactive. Here's an illustration:

Your monitoring system probably doesn't page you upon every '500' error code that is thrown. Instead, it is probably set to alert you if X number of error codes are thrown over a Y period of time. Whereas your monitoring system sees this period as one of "no alert needed," an SLO approach will give you a clearer picture of your risk by revealing that you are, in fact, burning tiny bits of error budget during this time.

For example, let's assume you have a large service that requires that you do your releases in a rolling manner across 50 pods. Although you've never heard complaints during your rollouts, your error budget shows burn during this time. Knowing this, you may be able to proactively improve reliability just by slowing down your pace of rollout across the pods. SLOs can even tell you when your reliability goals may be unnecessarily high or when you should ignore non-important false issues. Monitoring alone does not provide this kind of insight.

Monitoring Is Not the Best Language for Organizational Collaboration

Most monitoring systems are used by highly technical engineers, and a discussion about monitoring metrics can quickly make non-engineers mentally check out of a conversation. Therefore, monitoring data provides a poor basis for objective discussions with other stakeholders in the company. SLOs, on the other hand, provide a higher-level language of reliability that everyone in the company can understand and use. For example, If you're trying to communicate with your CEO, SLOs formalize the "squinting at metrics" exercise and elevate the discussion above technical jargon to business outcomes. Furthermore, SLOs are a language engineers and product managers can use to find agreement on threshold targets, prioritize, track progress, and develop valuable reports. (Read more about that here.) By providing a common language, SLOs contribute to healthier cross-team collaboration.

As I've said, monitoring is great. Monitoring is critical. But monitoring data is only a piece of the picture. SLOs complete the picture by focusing your attention on what truly matters to your customers.

That's what the Nobl9 platform is all about. Nobl9 collects the key metrics from all of your existing monitoring systems, including Datadog, New Relic, Prometheus, Lightstep, and more. We take these service KPIs (aka service level indicators, or SLIs) and do all the mathematical computations to establish the target percentages (SLOs), calculate your performance, and track error budgets.

Sure, some large organizations like Google can afford to devote a whole team to doing all the custom math and coding required to build custom SLO calculators in house. But most companies would prefer not to build these math machines themselves, particularly when there is already on the market an excellent product built by SLO experts.

Visit https://nobl9.com/platform/ to give Nobl9 a try.

Creating Your First SLO: A Discussion Guide

Originally published on Nobl9.com on May 29, 2020
Author: Brian Singer

Congratulations! You are ready to sit down with your team and establish your first Service Level Objective (SLO).

You might be wondering where to start. Here's an outline of how you could approach your first SLO-setting discussion with your developer and operations teams:

1. **Share a user story.** Suppose you have an e-commerce user story that says the user expects to be able to add things to their cart and immediately check out. Your user has a certain latency threshold for checkout, and when checkout takes longer than that, your user gets upset and abandons their cart.

2. **Phrase this customer experience issue more precisely as an SLO.** What proportion of users should be able to add items to their cart and check out within X amount of time?

3. **Identify and quantify the risks.** What happens if a customer isn't able to check out within that time frame? What does it cost when the SLO is missed?

4. **Brainstorm the risk categories together.** What are the things that can go wrong that would cause us not to be able to meet the SLO? Your team will respond with a wide variety of risks, likely including "our underlying infrastructure might go down," "maybe we pushed a buggy update," "we didn't anticipate so much demand all at once," and more.

5. **Ask "how could we mitigate these risks?"** When considering the resources/costs required to mitigate the risk versus the cost of failure, what do you leave to chance and what do you take a proactive approach to? Use this information to determine the service level indicators (SLIs) you will use to measure and track your ability to meet the SLO.

As you might imagine, this can be a fairly involved discussion, and all the stakeholders need to contribute their perspective in order to have buy-in in the end. It may take a while to find agreement, but when you do, you will find that among the developers and operators there is much more genuine understanding of (a) what the customer wants, (b) what new product features will truly cost, including operational support and capacity, and (c) how to prioritize and make customer-centered decisions when tradeoffs are necessary.

We'd love to hear how your first SLO-setting discussion goes. Let us know on twitter @nobl9inc.

Objective Launch Decision Making With SLOs

Originally published on Nobl9.com on October 12, 2020
Author: Brian Singer

If you are a product manager or tech lead, you typically face three key decision points in the process of issuing a new software release:

1. What's in the release?
2. Go or No-go?
3. Who takes responsibility for ongoing operations?

SLOs can play a helpful role in each of these decisions.

Using SLOs to Decide What's in the Release

In each release, you can work on features, bug fixes or technical debt, but you can't do them all at 100% each. Instead, each of these activities receives only a piece of your resource pie. Your challenge is figuring out how to allocate scarce resources in order to have the biggest impact for your customers.

What's the best way to plan what the engineers will work on sprint to sprint? Some product managers talk to all the stakeholders and let them vote. Some do squeaky-wheel-style release planning, focusing on the issue favored by the vocal minority. And other PMs like to chase the new shiny features, especially when a big customer has asked for them. Unfortunately, product managers can face all sorts of biased pressures that are not in the best interest of the business.

The danger of succumbing to those biased forces is that they tend to deprioritize—or even ignore completely—the very important but often invisible elements of release planning such as reliability, cost efficiency, cost to serve, security, privacy, compliance, and performance. Raise your hand if you've ever skipped these sections in a PRD? These things are not necessarily visible… until they are! And then you have a problem.

There's a better way: SLOs offer visibility and objectivity into these "silent assassins" of customer satisfaction. SLOs help you prioritize your release mix and avoid potential crises by calling to your attention the very important but often invisible "technical debt" issues that need to be addressed in your release cycle.

You may not even know exactly what technical debt you have until put your ear to the ground to hear the train coming. As your team starts to listen to their SLOs— setting objectives, measuring, and examining your scorecard—SLOs will alert your team to the exact technical debt issues that need your urgent attention. SLOs make the invisible visible.

Another way of looking at it is that SLOs give the important task of retiring technical debt or improving performance a respected seat and equal voice at the decision-making table right alongside new features. These aren't necessarily things that are easy to present in a customer roadmap meeting unless you have the data to back it up as well.

Keep in mind that SLOs should not be used to create hard-and-fast rules. At one of our recent SLO Bootcamps, a company described how they had set a two-thirds to one-third rule of devoting two-thirds of each release to features and one-third to retiring technical debt. As it turns out, this rule was actually causing them to focus *too much time* on technical debt when they weren't having reliability issues to warrant it, and this "over-engineering" of reliability was stifling innovation and agility. On the other hand, there's also a temptation, particularly among startups, to take the opposite approach: "move fast and break things." Taken to extreme, this disregard for reliability can be just as dangerous to your business, especially if your product is in a market that places a premium on stability.

The point is, don't use SLOs to create draconian laws that defeat their purpose; rather, use SLOs as a framework to bring feature velocity and reliability priorities in balance, and help teams make the right decisions.

Using SLOs to Make Go/NoGo Release Decisions

In most cases, SLOs may guide you to simply cut the lowest-priority feature from your feature stack to make room for reducing some technical debt.

In extreme cases, such as when you experience a serious reliability issue in the previous period, you may take the approach that Google advocates and actually block your team from releasing any new features until the technical debt is resolved.

Even in this era when continuous delivery is the favored approach to software development, Google SREs have been known to intentionally extend the release staging time for four or five days just to gather sufficient reliability metrics to guarantee a reliable launch in its network, even when concern about the software itself was relatively low. At Google, there's no doubt that reliability and customer satisfaction are paramount. But not everybody is Google.

It's important to keep in mind that Google, one of the early innovators in SRE, is at the far high end of the spectrum of SRE best practices. Google knows what's possible for their organization and, frankly, with over one billion active monthly users, they have a lot to lose if they don't deliver. (For most of those one billion users, Google *IS* the internet.) Google has a really, really high internal expectation about what reliability can look like, and that colors their perspective of what a team should do to deliver a service. Google also has a well-honed, universally accepted SRE culture that invests in operational readiness. Google PMs expect to put effort into reliability, even before release planning begins, and Google SREs are perfectly comfortable drawing the line to protect the company against risk.

If you're not Google, and you don't yet have a culture fastidiously devoted to reliability, you need to adjust your approach to what's right for your organization. What is your highest reliability capability? Keep it in the realm of possibility. (But, yes, you still need SLOs.)

Using SLOs to Establish Operational Responsibilities

When it's time to move a product release from development to production, the "elephant in the room" is often the question "who is going to take responsibility for production?" SLOs are a great way to guide the decision of who takes the pager.

Of course, companies can organize their development and operation teams in many different ways. Some may take a "you-build-it-you-run-it" approach, which means the development team is directly incentivized to run the application in production efficiently. Many organizations divide labor between those who build stuff and those who run stuff. This approach might be a holdover from "the way things have always been done," or it might be a strategic decision based on the skillsets of existing personnel or an attempt to optimize team interests and focus.

Regardless of why or how engineering teams are organized, SLOs are great for creating alignment during hand-offs to set clear expectations of reliability and to set a threshold for the minimum reliability for a given product or service.

For example, suppose the dev team comes to the ops team and says, "This is mission critical; please run it at five nines." SLOs give the ops team cause and justification to push back: Have you been able to achieve that historically? Can we break down the reliability goal a bit and understand which pieces really need five nines? Is the company prepared to deal with the necessary costs and slowdowns to meet that goal?

SLOs also are invaluable after handoff. Perhaps a team has already handed off a release, and a series of reliability issues have cropped up over time. The operational team (or the SRE team, same difference) can use SLO data to open a conversation about how to handle the issues and the shrinking error budget. Maybe the dev team joins the rotation to gain operational empathy or even "takes back the pager" for a while. Another option might be putting new features on hold or mostly on hold until reliability goals are met. Yet another approach would be to have a serious discussion about changing the reliability goals!

In summary, SLOs are ideal for helping product managers and tech leads address the three key decisions that need to be made with each release:

1. What's in it? SLOs help you give technical debt its due consideration.
2. Go or No Go? SLOs help you set the reliability bar at the proper height for your organization.
3. Who's in charge of ops? SLOs help your dev and ops teams align before, during and after hand-off to set reliability objectives and resolve issues in production.

SLOs should be a part of the logic and decision-making process of every product manager as well as the development and operation teams. In fact, SLOs should be an integral component of the "blood stream" that runs through your entire organization from customer engagement to service delivery.

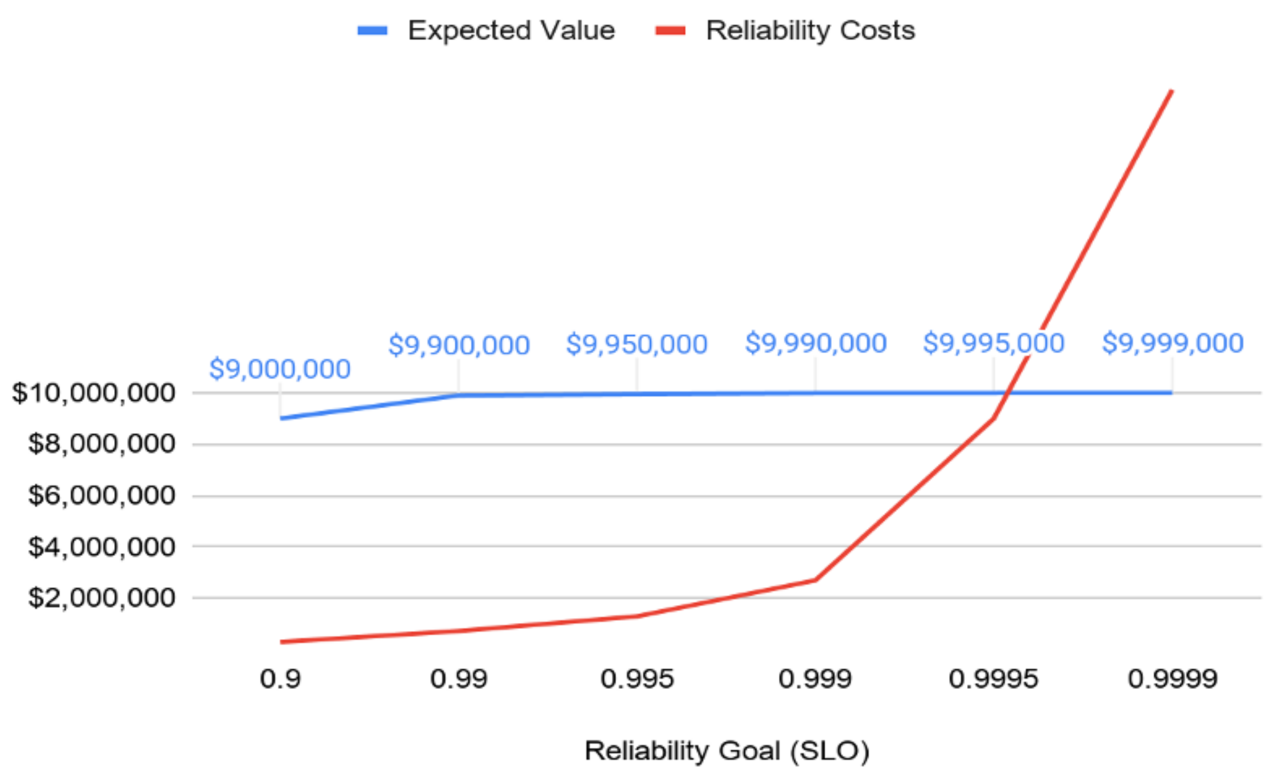
SREs: Stop Asking Your Product Managers for SLOs

Originally published at devops.com on October 30th, 2020
Author: Kit Merker

One of the fundamental premises of software reliability engineering is that you should base your reliability goals—i.e., your service level objectives (SLOs)—on the level of service that keeps your customers happy. The problem is, defining what makes your customers happy requires communication between software reliability engineers (SREs) and product managers (PMs) (aka business stakeholders), and that can be a challenge. Let's just say that SREs and PMs have different goals and speak slightly different languages.

It's not that PMs fail to appreciate the value that SREs bring to the table. Today, in the era of software as a service, features such as security, reliability and data privacy (which used to be considered "non-functional requirements") are respected as critical features of the service-product a SaaS company delivers. Modern application users and customers of software services care *a lot* about data privacy, cybersecurity and uptime; therefore, PMs care, too. In fact, it's not uncommon to see these features touted prominently on a company's website (I saw one today claiming "99.99% uptime for every customer!") because the folks in marketing know that customers are making purchasing decisions based on whether the company can deliver reliability, speed, security and performance quality. So, yes, PMs *do* care.

The difficulty lies in the negotiation between the PM and the SRE about the value of providing that level of service and the cost associated with it:



The people and infrastructure costs associated with providing the next level of reliability increase geometrically for a host of reasons, but the expected value to a $10 million business of losing out on that one in 1,000 transactions versus one in 10,000 is only $1,000.

But translating “customer happiness” into actual objectives derived from observable metrics is easier said than done, particularly for companies that are just getting started with SLOs. Let’s face it, all the talk about “nines” can be difficult to conceptualize. Fortunately, there’s an easy way for PMs and SREs to get on the same wavelength.

Engineers, stop asking your PMs for SLOs! Instead, convert user stories into reliability goals.

User stories are a great way for PMs to express a customer’s experience and expectations to the engineering team. SREs can then translate the user stories into SLOs in a few simple steps.

What is a User Story?

User stories are narratives that describe how a particular type of user engages with the service. The first-person plotline might go like this: I am *this type of person*, and *this is what I need and expect* to be happy, and *if I’m not happy, this is what it’s going to cost you.*

Note: In this user story, we’ve taken the liberty of including empathy for the reliability expectations of the user.

Retail example: I’m a shopper at your online store on Black Friday, and I want to take advantage of your Black Friday sale. I want to log on, peruse your Black Friday deals, pick the products and features I prefer, and make my purchase in just a moment or two. If it takes too long to log on, or if I get hung up in the system, I’ll easily get frustrated and move on to other retailers. I don’t want to miss the deals!

An SRE can take that user story, ask the PM a few questions to gather just a few more critical bits of information, then do some simple math to derive the SLO for your Black Friday.

Ask Your PM These Questions

To determine the reliability goals for a user story, you need to add three pieces of context:

Step 1 – Identify the Top Critical User Transactions

First, determine the relative priority or criticality of the transaction being successful. Not all transactions are created equally, so we need to understand what kind of transaction we’re talking about here. For example, browsing a catalog page (which could be refreshed if unsuccessful) is different from the shopper’s checkout experience.

Retail example: We have several transactions in a retail business, including:

1. Inventory catalog search.
2. Order fulfillment.
3. Payments processing.
4. Price and discount.
5. Checkout experience/shopping cart abandonment.

For our example, let's develop the SLO for payments processing, which is a throughput metric—the total number of transactions a retailer can handle in a given period of time.

Step 2 – Estimate the Volume of Transactions at Various Times

Now that we know the transaction we're talking about (payment processing throughput), we can estimate the rough volume of transactions or user interactions that will occur in each of those categories, rounded to the closest order of magnitude (see Fermi Estimation). How much of your business happens during this time? Do you have a critical time of day? Do you have a critical season? Do you have a critical day(s) of the year?

The same transaction at a different time may be more or less valuable. Think about an ad playing during the Superbowl or stock prices while the market is open. Or when your CEO is giving a critical customer presentation. Timing matters.

Retail example: Our hypothetical retailer has critical periods in the holiday shopping season and a critical day called Black Friday. Typically, 4% of annual business happens during Black Friday weekend, and 16% of annual business happens during the remaining holiday season.

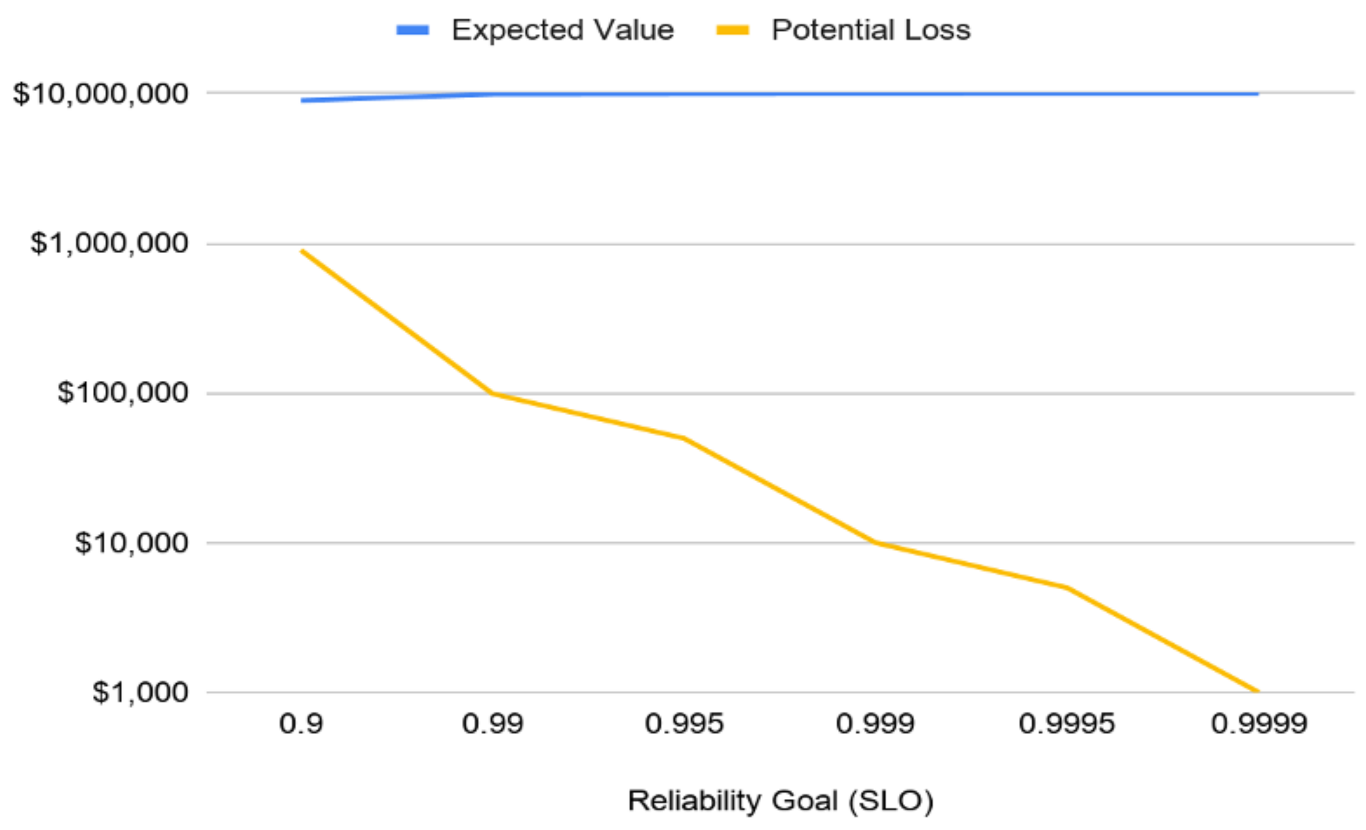
Adobe Analytics estimates that sales for the full weekend (Thanksgiving through Cyber Monday) is 20% of total revenue for the full holiday season. According to the National Retail Federation, 2019 holiday sales were $730.2 billion, accounting for 20% of annual sales.

Step 3 – Estimate the Business Value of Transactions in Each Circumstance

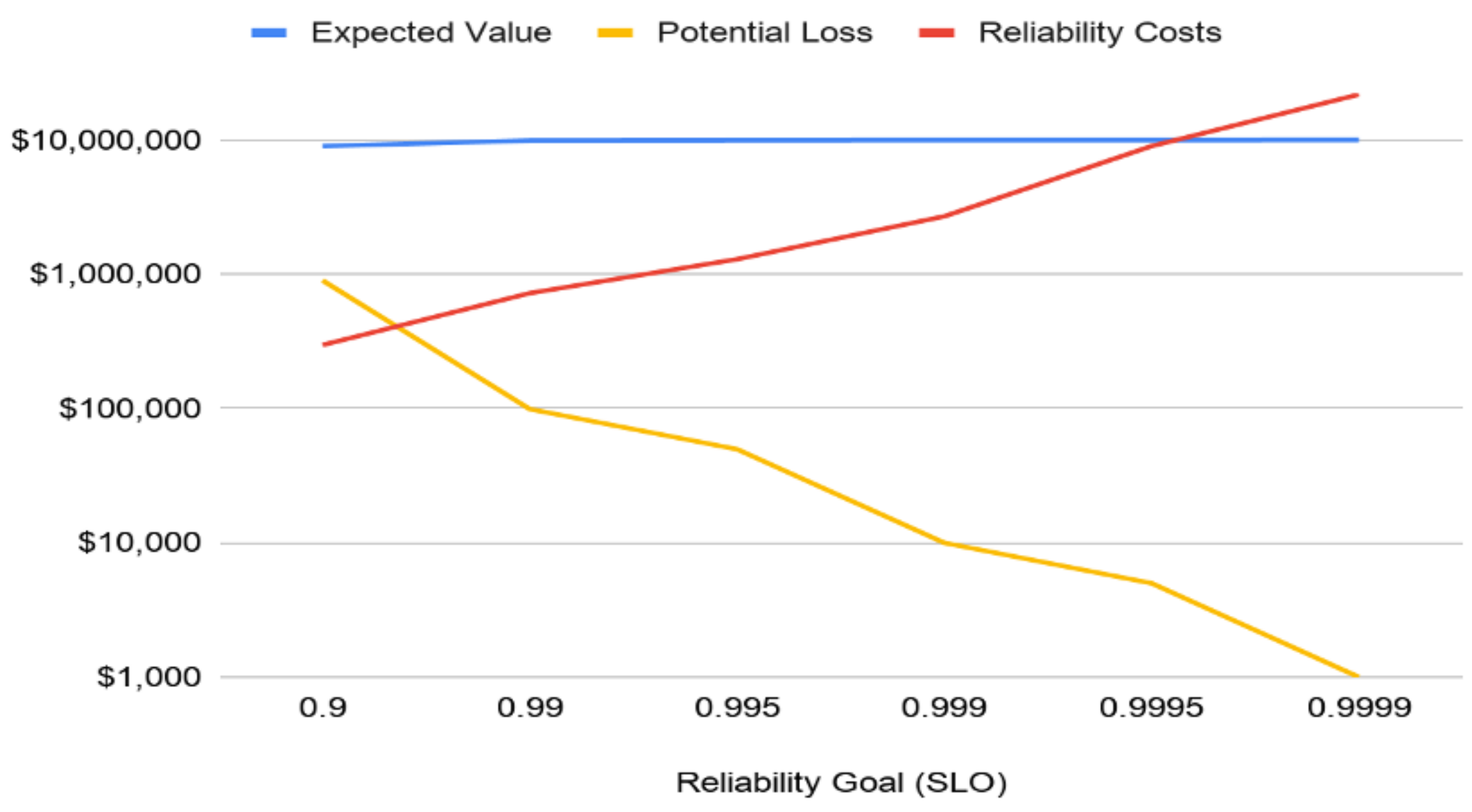
Finally, we can now understand how the reliability of this transaction affects business outcomes. This could be the risk of lost revenue or perhaps a brand reputation metric such as reduced CSAT. Now you have a pretty clear picture of the business impact of unreliability.

Retail example: If our imaginary retailer has revenue of $10 million, it would make $400,000 of that during Black Friday weekend, and $1.6 million during the rest of the holiday season. On Black Friday shopping carts are worth $100, $80 during the holiday season and $50 the rest of the time. There are 800 transactions per day on Black Friday, 625 per holiday season day and 488 any other day.

Now that you have this input, you can start to calculate appropriate SLOs based on the value at risk and the cost to meet that reliability goal.



If possible, you can estimate the cost to deliver each level of reliability. Consider not only the cloud computing and direct operational costs, but also the reliability investments in testing, performance, security, chaos engineering, release management, rollback, monitoring, on-call, automation, etc., that would be required to achieve the reliability level at each point. By plotting this cost (at least to the order of magnitude) you can find a break-even point where the value at risk and cost of reliability crossover. This is the business-justifiable SLO for these transactions.



Plot the value-at-risk as a function of **value** x **volume** x **SLO**. The value-at-risk will decline sharply as the reliability is increased; however, the cost to achieve this reliability also increases exponentially. It's simply not worth investing more than that at-risk amount for any given user story, so the SLO determination is clear. Note that the SLO may not be achievable for the team currently, but you have a precise idea of what matters to the business and the break-even point of your reliability investments.

Instead of asking PMs to "speak SRE," span the communication gap by using the common language of user stories to build business-cogent SLOs.

Guest Chapter: How to sell SLOs to Engineering Directors

Originally published at medium.com on December 18, 2020
Author: Thomas Césaré-Herriau

***Thomas Césaré-Herriau** is the technical lead of the Observability team at Brex. His main interests are empowering other teams to build well-designed, reliable, and highly observable systems by providing powerful abstractions and primitives.*

We recently started our Service Level Objectives (SLOs) journey at Brex and thought it would be valuable to share our learnings with this community. This blog is a redacted internal memo that aimed to familiarize SLOs with its audience, explain the value of an SLO culture, and describe how we would implement and roll them out.

We hope you find it helpful.

From: thomas@

To: engineering-directors@

Date: 06/15/2020

Subject: S.L.A.E.I.O.U — An SLO Strategy proposal

Executive Summary

To successfully scale with our customer base there are three aspects that are critical to maintaining and further improving trust;

1. improving reliability;
2. controlling risk; and
3. increasing iteration speed.

All aspects that are only achievable if we have an accurate way to measure our services. This is best accomplished by defining and capturing Service Level Objectives (SLOs).

With SLOs, we will be able to estimate the business impact of our services' reliability, and decide which services can be iterated upon more quickly, by targeting lower reliability, thus accurately understanding and controlling the risks taken.

Background

In order to achieve the above we know we need to;

- distinguish High Reliability from High Velocity projects, and iterate as fast as the associated risk tolerances for each allows.
- continue to increase Brex's overall reliability as we expand our customer base.

Despite early Brex being built as a modular monolith, we have started undertaking several efforts to decouple our services, improve the reliability of our core systems, and increase overall developer velocity. However a key component of being able to achieve these goals, is to be able to track how well our systems are behaving from our users' perspective in order to make data-backed decisions based on the type of project.

This is where SLOs come in.

In a nutshell, an SLO is an agreed upon target of a quantitative indicator determining how well our service is behaving from our users' perspective.

Glossary

Critical User Journeys, SLIs, SLOs, SLAs and error budgets are key concepts from Software Risk Management introduced by Google SRE and widely used throughout the technology industry. Here are short definitions:

Critical User Journey: any user facing core feature, for example:

- Credit card authorization
- Displaying transactions in the dashboard
- Receiving a transaction approved SMS

Service Level Indicators (SLI) are a measurement that are used to determine if a system is healthy or available from the perspective of the user (i.e "Percentage of succeeded requests" or "Latency of an action")

Put in other words, it's a quantifiable measure of reliability

Service Level Objectives (SLO) define a healthiness or availability goal for a system based on a

SLI (i.e "99.5% of requests succeed" or "99% or requests take less than 200ms")

Service Level Agreements (SLA) generally define a legally binding contract with a paying customer about a set of SLOs (i.e "If less than 99.5% of requests to your service succeed, a refund will be issued")

Error Budget, derived from an SLO, is the amount of time per period (usually 7, 28, 30 or 90 days) during which a service can violate its target SLO.

A 99.9% SLO has a 0.1% error budget over the period used.

Service Scorecard: a methodology to quantify the operational quality of a service. It generally uses checks or measures such as:

- Does the service have an oncall rotation assigned to it?
- Does the service have alerts defined?
- Does the service have a dashboard?
- Was the service's architecture reviewed and approved?
- Test coverage etc.

Goals

- All Critical User Journeys have defined and associated SLOs.
- SLOs are used as part of OKRs and project planning processes.
- High Reliability / High Velocity projects are accurately described with corresponding high / low SLOs.
- Supporting services that do not directly expose a user-facing feature also have defined SLOs.
- As a Brex engineer, I can instantly understand the health of my systems.
- As a Brex employee, I can instantly understand if our product is operating as expected, and if not, what areas are experiencing issues.
- As a Brex customer, I can access a status dashboard that clearly describes the current status of Brex products.

Why SLI, SLO and SLA matter

Business Impact

Capturing SLOs across our main Critical User Journeys will allow tying reliability and availability to a dollar business value. As our business grows, a non-increasing availability over time will have an increased business impact.

Let's take a look at a simple example, an online ecommerce business, and the reliability of its payment flow that happens once a customer has put items in their cart and are ready to pay for their order. The SLO is defined as 99% of the payments initiated are successfully processed. If we don't process the payment, the customer will get an error and for the purpose of this example, we assume they will give up on their order.

- Initially, we process around $1M of orders every 30 days.
- For the purpose of the exercise, we assume they are distributed evenly over those 30 days.
- We expect to not handle properly 1% of them, so that's $10,000 of revenue every month.
- It may not be worth dedicating engineering resources to "add a 9".
- Now, if we were to grow our GMV to $10M a month we would then be losing $100,000 in revenue every month.
- Probably time to dedicate or hire engineers for this problem!

The part that is not captured in this example is the customer trust that we erode every time we decline a transaction inaccurately. This is harder to quantify, but will definitely have an amplified impact as we expand our customer base.

Tactical impact

The goal of implementing SLOs is to make informed decisions about what systems we invest engineering resources in, and how.

As we can tie a specific SLO to a business value, this will allow Product Managers and Engineering Managers that are planning projects to know how much resources to allocate to tech debt / improving the reliability, and how much to dedicate to feature development.

This is also going to be key to clearly, and with data, distinguish High Reliability and High

Velocity services:

- High reliability services require high SLO (>=99%)
- High Velocity services are OK with low SLO (<99%)

During the design phase, SLOs and SLAs are keys to be able to understand the availability implication of different external services (AWS RDS, AWS DynamoDB, Amazon MSK) as well as internal services (Events Infrastructure, other dependencies). SLOs clearly define the assumptions engineers can make when depending on other systems. Based on the level of availability one new service should achieve, using an SLO/A rich environment will facilitate the decision.

Operational impact

"It's impossible to manage a service correctly, let alone well, without understanding which behaviors really matter for that service and how to measure and evaluate those behaviors"
— Google SRE

Traditionally, alerts and customer reports are used to understand whether services are functioning or not. While this helps detect failures, it does not indicate whether over time we are serving our customers well. Alerts help us mitigate risks, not measure it.

Properly defined SLOs will increase developers velocity by increasing their confidence in releasing new changes, and interacting with other services, for the following reasons:

- Clearly defined SLOs will make sure that systems have well defined capabilities and track the performance of those.
- SLOs help to ensure that performance of the underlying system stays consistent as the system evolves.
- SLOs are similar to testing, but at runtime in production: it measures whether our Critical User Journeys are behaving as intended.
- SLOs derived alerts quickly and accurately capture user-visible issues.
- By setting two levels of SLOs (one external and one, tighter, internal) we can prevent issues from becoming user-visible.

By capturing in real-time how well our systems are behaving according to our Critical User Journeys, it will allow us to provide detailed status pages to internal teams that will help quickly understand a reported user issue.

Plan

The below has been modified to provide more generalizable information:

In phase one, focus on one well-known high reliability team and one well-known high velocity team to define and implement an initial set of SLOs. This is a great opportunity to build rapport and develop tooling and documentation to help subsequent teams.

Remember to add a SLI/O section in your design doc templates to create self-reinforcing processes!

In phase two, identify a subset of services/features to have SLOs defined and work with management to prioritize this work. The output of Phase 1 should help other teams self-service this work. This will continue to increase the organization's knowledge and practice, slowly building the culture.

In phase three, require all services to have SLOs defined before launching to production.

Alternatively, a forcing function such as a Service Scorecard can be used.

Why now?

SLOs are a cultural shift. It is about understanding our systems from our users' perspective, ensuring that what we build provides the quality of service our users deserve. It is about measuring and tracking what matters.

All teams have an understanding of what their services do, SLOs are about formalizing them. It is easy to start, hard to get it right at first and it requires iterating until we get to an understanding of what to track and how, and how an SLI relates to the perceived quality of service from our users.

As such, the earlier we start defining and using SLOs the better.

That's it! A version of this memo was used to successfully gain buy-in and schedule time for creating SLOs across the engineering organization. In the following posts, we'll talk about the progress, obstacles, and learnings of our SLO journey.

Key takeaways

- Use the language Directors care about: be business centric.
- Explain the value of SLOs at different levels: strategic, tactical and operational.
- Ensure you are able to provide close support to early adopters, and go for a gradual rollout with a few pilot teams initially. Be customer centric.
- Rolling out SLOs sooner than later will allow a culture of reliability and risk management to slowly take its roots, and to build the foundation for successfully managing massive growth of your customer base.

SLOs Are Good; SLOs for Defined Customer Segments Are Better

Originally published on Nobl9.com on September 28, 2020
Author: Alex Nauda

SLOs are a powerful tool to increase your engineering organization's maturity in terms of reliability and risk management: use them!

We in the SRE world often speak in generalities about "customer happiness" and how SLOs can help us find that ideal balance between software reliability and the velocity at which we release new features. To be sure, for many of our conversations, it's expedient to refer to "the customer" as a homogeneous entity, as if we can meet a certain reliability goal and the "collective customer" will be happy. But in the real world, there's no such thing as the homogeneous customer, and there's no such thing as a one-size-fits-all SLO. We need to be able to account for customer heterogeneity in our reliability monitoring.

- Fact: Customers differ in where, when, and how they use your infrastructure.
- Fact: Customers differ in how important their workloads are for internal and external end-users.
- Fact: Customers differ in how important their business is to your company.

At Nobl9, our premise is: SLOs are good; SLOs for defined customer segments are better.

We have been thinking a lot about this and how best to go about accounting for the reality of diverse populations when designing reliability monitoring. Here are three ideas that can help you take SLOs to the next level by segmenting user experiences:

Imagine how powerful it would be if the SRE team could "see" the value of individual users.

1. Understand user experience at the individual user level.

Just because you have a 0.1% failure rate (i.e., 99.9% uptime) doesn't mean 0.1% of users are experiencing it. So how do you know which users experience that failure? Or, to take it a step further, how do you know what any given user is experiencing with your service at any given moment?

Techniques are emerging that will help us measure experience at the user level and define what actually happens for a given user. For example, you can read about Google's approach to windowed user-uptime here.

Another possible approach is using user IDs (or user emails or user names) to map customer relationships. Multiple user IDs may map to a single organization or a single user

ID may map to multiple organizations. Either way, by understanding these connections, we are able to isolate key user experiences.

Here's a simple illustration: let's look at a video streaming app. If you know that 0.1% of video playbacks are buffering slowly, you could drill into that and see how a specific customer company is experiencing the delay. Or, further, you could potentially see how a project, team or individual customer user is experiencing the service–an invaluable insight when that individual user is the CEO trying to show a video during a key investor presentation!

The point is, we need to explore ways to slice and dice our SLO metrics for unique users.

2. Incorporate business intelligence into an SLO-based "triage system" to assess and prioritize unique user needs.

The triage concept is familiar to the DevOps teams of most B2B SaaS organizations. ("Should we wake up an engineering director now?") But at Nobl9, we have dreams of doing triage better. The reliability-focused triage system we envision would take into account customer segments and incorporate business intelligence to reveal the unique facets of user relationships.

Imagine how powerful it would be if the SRE team could "see" the value of individual users. For example, suppose User A appears to simply be testing a trial account, but in reality User A is also a huge potential customer that your sales team has been trying to close for two years? That critical information is probably reflected in Salesforce or some other CRM system, but totally unknown to the SRE team. To apply SRE in its most potent form, important business insights about individual customers need to be available to the team in real-time.

The closest thing we have to that ideal triage system today is when companies segment their customers based on contract value and SLAs. The value of an existing account is a good place to start, but it shouldn't be the only factor considered in the triage system.

3. Emerging technologies will enhance our ability to customize reliability metrics to unique user populations.

The idea of prioritizing workloads isn't new. IT operators are accustomed to making workload adjustments to protect the interests of internal and external customers. For example, we can isolate specific accounts or tenants onto a hardware environment that is more performant and reliable than the general population. Or we can dynamically move customers from fast to slow to even slower lanes of traffic to accommodate workload types, such as a large batch upload that need not be serviced immediately, while preserving performance for other customers whose workloads are more time sensitive.

What's new today is the idea of using SLOs to guide and automate these workload prioritization and infrastructure decisions, and that's what Nobl9's technology is all about.

In the future, we foresee advancements in routing, load balancing, ingress software, service mesh, and other critical infrastructure services that will give us richer features to isolate, track, and proactively adjust service levels using SLOs as our guide. I think we're just scratching the surface of what is possible.

So, let me ask you: Where do you stand today in your ability to identify your most valuable customers and isolate their unique experiences with your service? At a minimum, I hope I've challenged you to think more rigorously about how to segment your users with your own user and customer account data. I hope I've also convinced you to take the next steps toward applying SLOs to segments of your users. You can start by segmenting customers using financial data, then layer on the more subjective business intelligence you have, such as projected LTV. Next, pick a segment, put baseline SLOs in place, and begin turning knobs to fine-tune your reliability efforts.

Remember: SLOs are good; SLOs for defined customer segments are better. I can assure you that we'll continue to explore this topic at Nobl9 and share our insights here in this blog on a regular basis.

Are you ready to take your reliability to the next level?

It is becoming increasingly more critical for teams to create a culture focused on the customer experience. SLOs complete the picture by focusing your attention on what truly matters to your customers.

Now that you have learned what a CEO should know about SLOs, has this Ebook given you food for thought?

Join our two-day SLO Bootcamp for a hands-on workshop to help teams define Service Level Objectives (SLOs) for their services.

The half-day Bootcamp includes short educational lectures on reliability and SLO methodology.

The Bootcamp offers:

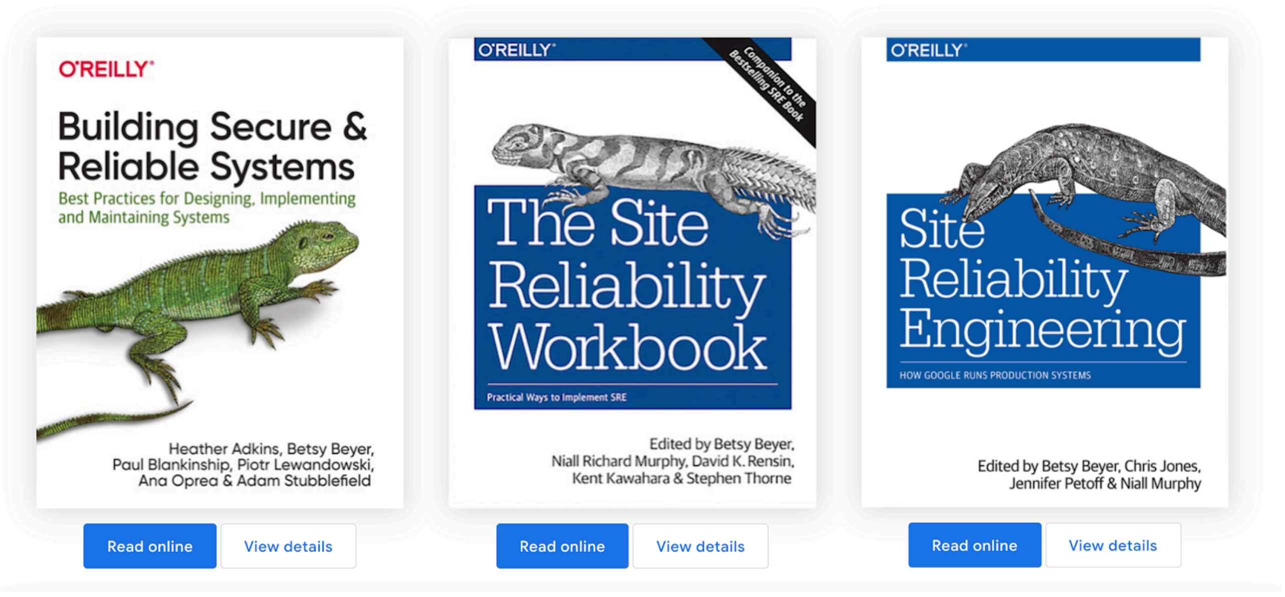
- Hands-on Education
- Small groups work with designated
- "Service Owners" who represent real
- services for organization
- Setting clear Error Budget Policies

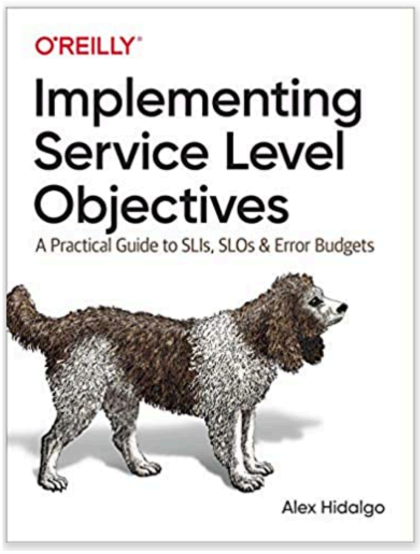
Sign up today by contacting us at bootcamp@nobl9.com

Further Reading

- Google SRE Books
- Google SLO Report
- Implementing Service Level Objectives
- SRE Blueprint: Creating and Fulfilling SLOs for Optimized Business Outcomes (IDC Report)
- How To Optimize Gross Margins For Digital Services
- Three Ways Business Development Leaders Can Help Their CTOs
- Three Steps For Measuring Customer Experience In A Digital Business
- Nobl9 Ties Business Goals to Observability Da

SRE Books


O'REILLY
Building Secure & Reliable Systems
Best Practices for Designing, Implementing and Maintaining Systems
Heather Adkins, Betsy Beyer, Paul Blankinship, Piotr Lewandowski, Ana Oprea & Adam Stubblefield
Read online
View details
O'REILLY
Companion to the bestselling SRE book
The Site Reliability Workbook
Practical Ways to Implement SRE
Edited by Betsy Beyer, Niall Richard Murphy, David K. Rensin, Kent Kawahara & Stephen Thorne
Read online
View details
O'REILLY
Site Reliability Engineering
HOW GOOGLE RUNS PRODUCTION SYSTEMS
Edited by Betsy Beyer, Chris Jones, Jennifer Petoff & Niall Murphy
Read online
View details


O'REILLY
Implementing Service Level Objectives
A Practical Guide to SLIs, SLOs & Error Budgets
Alex Hidalgo