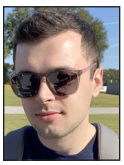
It's an SLO World

What Theme Parks Can Teach Us about User-First Reliability

JAIME WOO AND EMIL STOLARSKY

Jaime Woo is an award-nominated writer and has been published in the *Globe and Mail*, *Financial Post*, *Hazlitt*, and *The Advocate*. He spent three years as a molecular biologist before working at DigitalOcean, Riot, and Shopify, where he launched the engineering communications function. He's spoken at SXSW, IA Summit, SREcon Americas, and SREcon EMEA, and was a guest lecturer at the University of Toronto's Rotman School of Business. He is co-founder of Incident Labs and is co-authoring the forthcoming book *SRE for Mere Mortals*.
jaime@incidentlabs.io



Emil Stolarsky is a Site Reliability Engineer who previously worked on caching, performance, and disaster recovery at Shopify and the internal Kubernetes platform at DigitalOcean. He has spoken at Strange Loop, Velocity, and RailsConf, was a program co-chair for SREcon19 EMEA, and is a program co-chair for SREcon20 Americas West. He has guested on the podcasts InfoQ and Software Engineering Daily, and contributed a chapter to the O'Reilly book *Seeking SRE*. He is co-founder of Incident Labs and is co-authoring the forthcoming book *SRE for Mere Mortals*.
emil@incidentlabs.io

In an always-on world, predictable reliability is paramount. Service level indicators (SLIs) and objectives (SLOs) are cornerstones in site reliability engineering (SRE) for purposeful reliability. SLIs are chosen measurements that act as signals for achieving your reliability goals; SLOs are the targets for SLIs. User-first SLIs and SLOs are the gold standard, and we use the concept of theme parks, those paragons of complex systems optimizing for user happiness, to demonstrate examples of strong SLIs and SLOs in contrast to useless ones.

The massive, iconic theme parks of Orlando, Florida, are impressive for children—the rides, character actors, and sights synthesize into magical, larger-than-life playgrounds. It was surprising then to realize how much more impressive the spaces become when revisited as adults. The infrastructure that manages hundreds of thousands of visitors daily, the attention to detail across the "lands"—even in places where people might not immediately notice—evoke awe and appreciation for the levels of planning and effort.

The lessons for site reliability engineers from theme parks are not immediately obvious, until you realize that SLOs are rooted in asking what level of service must be provided to keep users happy. And where else could you glean lessons about how to engineer for happiness than at the so-called happiest place(s) on earth?

Useful SLIs

Let's begin with how to find a useful SLI. A useful SLI must contain the following four parameters:

- Relate to the experience and/or satisfaction of your user
- Use a measurable quantity related to your service level
- Be as specific as it can be
- Provide enough information to be actionable

Similar to the massive infrastructure that is behind what appears as simple user-facing experiences, underneath the colorful, playful facades of theme parks are subterranean levels where workers manage the infrastructural and logistical components of the park, including electrical operations, transporting character actors, waste removal, deliveries, and food service [1].

Visitors rarely think (or even know) about these hidden parts—and that's the preference of theme parks so as not to ruin the illusion. The only thing that matters is the experience visitors paid to have, and everything that is out of view exists only in support of that experience.

Take waste removal: should trash begin to pile up around the park, visitors would complain about seeing garbage on the park grounds, rather than, say, faulty waste removal mechanisms 20 feet below them. And, theoretically, those mechanisms could break and guests wouldn't notice whether staff cleared the paths of trash often enough. So the amount of garbage on the floor is a stronger SLI than waste removal machinery uptime.

That doesn't mean ignore everything internal: instead, we acknowledge that something can be important yet not necessarily urgent. That ambiguity around urgency highlights the disadvantage in using such metrics as guidance for reliability: because it's subjective, it's more difficult to gauge reasonable boundaries around allowable downtime and, therefore, to create meaningful error budgets to justify any downtime. Anything that users interact with directly affects their ability to do what they need to do, and thus prioritizing work is clearer.

For example, take a database service with multiple replicas. It might be tempting to use uptime as an SLI, but there are many scenarios where uptime gets dinged but customers don't feel the impact. For instance, if a single replica goes down, traffic won't be affected. Instead, a more effective SLI would be to track read-query success rates, which are necessary for customer requests to be successful.

From SLIs to SLOs

Upon determining SLIs, you have to assess the right target SLOs. From our theme park example, we've figured out that guests would be unhappy with trash everywhere. Now, we want to know what their threshold is, based upon their needs and expectations.

With small piles of garbage everywhere, the park technically remains operational, but it would be a poor experience for guests, potentially discouraging them from returning or even asking for refunds. On the other hand, ensuring no piece of trash stays on the ground for longer than a few minutes would be an excessive waste of resources. How then to choose the right level?

Luckily, engineers need not—and should not—do this alone. Different business units across the organization will have their own insights into users, and when site reliability engineers work with teams like support, engineering, and product and bring those insights together, you're likelier to have meaningful SLOs. At Disney World, you're unlikely to see trash on the ground for longer than 15 minutes, as that's the interval, in crucial locations, at which trash in bins get sucked into an underground automatic vacuum collection (AVAC) system and transported away.

With our example SLI of read-query success rates, after discussing with other business units, we may learn that users typically notice degradation in the service when fewer than 95% of read-queries succeed over a period of 30 minutes. Waking up engineers the moment any read-query fails would be premature, but we could set a slightly more stringent internal SLO that once read-queries drop below a 97% success rate, alerts get sent out.

What Is the Experience?

With the need to be user-focused firmly established, we can move from what users see to what users experience. The distinction between the two is that one measures what users interact with, and then the second looks at those interactions and translates them into their meaning. In the field of UX, they understand the distinction: "While you cannot directly *design* a person's experience of a product, you *can* take steps to ensure that their experience is a positive one by employing a user-centered design process," writes Matt Rintoul, experience design director of creative agency Say Yeah! [2].

At this point, we should make a vital distinction: who your users are matters. For this article, we focus on human users rather than programs that use your service (although users can be both, depending on which part of the service each touches).

Thinking about how different users interact with your system is a useful exercise. The response times from programs are more reliable and faster than with humans. You can also tell a program to attempt a request again in five minutes. Unlike machines, humans perceive things relatively, something we'll return to later in this piece. This matters because treating humans as rational actors, as economists do, can simplify things, but you need to be careful it doesn't oversimplify. Context matters.

What Constitutes a Satisfactory Experience?

Rarely is a service entirely down. Instead, individual components may lag or fail, and even with some parts of the experience deprecated there may be no change in a user's core experience. At a theme park, the food stands, for example, could be out of service, and the park could still run. If the restrooms all failed, however, it'd be a different story.

For a technical example, on a video-streaming service, there are many components to the total experience, from searching content, user-curated lists, viewing history, and playing content. Each component can be mapped to see if it is running or not, and this matters because the components are weighted related to user satisfaction: if customers can still continue watching a film they were in the middle of then they will be happy even if they cannot amend their list of media to watch.

Specificity matters because when outages occur, or decisions around what work should be prioritized, you make best use of your limited resources by understanding which parts of the service matter most to customers. You can then also manage the number of things being tracked in dashboards to prevent information overload. An engineer needs to weigh the tradeoff of adding another SLI to monitor against the level of dissatisfaction users will have if it goes down.

Timing Matters

Just as components of your service are relative, with differing weights, this is also true for time: not every minute is the same. If your users don't notice an outage, should it count toward

your error budget? At theme parks, for instance, electronic gates require fingerprint identification for entry. If this system went down an hour before the park closed, while that's not ideal there's also the nontrivial question of who was impacted?

This isn't permission to ignore outages that happen during the off hours. You still want to know how often your service is going down, which provides a better way to understand the behavior of your system. But does your service truly need to be up 24/7? Are there periods when the service is lightly used? It isn't zero impact, but it has less impact, so do your metrics reflect this? Importantly, is a low-impact event worth the human capital of waking a team at three in the morning?

As an example, a food delivery service that solely works with restaurants on the East Coast of the United States: most restaurants do not operate between two to six in the morning, and perhaps the service has data showing that orders drop off after 10 p.m. and only revive at 10 a.m. for lunch orders. An SRE team could decide that alerting overnight for low-severity incidents isn't worth sleep-deprived and grumpy engineers and instead send pages in the morning.

Users Have a Multitude of Experiences

Rarely are users a homogeneous monolith. Instead, they are heterogeneous, each with their own (albeit, potentially overlapping) needs. In UX, the practice of creating personas acknowledges that users have different perspectives and rationales. When considering SLIs and SLOs, we should avoid blanket aggregation of users for the sake of simplicity.

Returning to the theme parks of Orlando, think about the different types of visitors: parents and guardians, children, aunts, uncles, grandparents, and adults without children. They speak different languages. They have different accessibility needs. They have different cultural perspectives. As a result, theme parks provide experiences to cater to the wide range of needs and expectations.

An example was the introduction of single-rider lines: rides often seat visitors in pairs and therefore can have unused capacity when groups have an odd number of people or for solo visitors. Worse, solo visitors would wait as long as large groups, even as they could see empty seats on the ride. By creating a line just for individual riders who don't mind sharing with strangers, the excess capacity can be used up—providing a quicker queueing experience for all guests.

Users of technical systems are just as varied. They can come from different geographic regions, be of different sizes, vary in their frequency of use, and so on. And aggregating them is just as pernicious. An example is when a company has their datacenter in North America, where the majority of their customers are. If the data is aggregated, a customer located in Eurasia facing subpar performance might not trigger an alert: the user may become unhappy, even if all SLOs appear to be met.

User Perception Matters

Unlike machines, humans perceive interactions based on their past experiences and attempt to create context based on what has happened: a machine might make several attempts to connect without those attempts creating any kind of storyline. This is less true for humans, where they build theories based on patterns, and it is at our own peril to ignore this fact.

We cannot, obviously, measure how users feel at every moment because it is intrusive and expensive. We also do not want to rely on users venting their frustration at customer support or online on Twitter either, because then it's too late. But we can start thinking about user perception as a factor in our SLOs and acknowledge that it plays a role if we are to be truly user-focused.

Perception is by definition subjective, sometimes in counterintuitive ways. An illustration comes from a phenomenon called paradoxical heat: when a person holds a warm pipe in their left hand and a cool pipe in their right hand, they sense painful heat, even if neither pipe individually feels unbearable. We are unaware of a directly analogous phenomenon for SRE, but a similar idea might be having two minutes of downtime, followed by two minutes of availability, followed by another two minutes of downtime.

This won't feel like four minutes of outage: anyone who has experienced spotty WiFi coverage will understand the oddly intense anguish that comes from intermittent connectivity. It can feel worse than not having Internet access at all, because it robs you of your sense of control over the situation: should you keep trying or do something else? Not knowing whether the next outage will be in a minute or not at all can be very frustrating. So we can't just look at the raw data itself but have to also think about how that data represents experiences. Four one-minute outages alternating over an eight-minute period may feel worse than a continuous four-minute outage.

Perception also plays a role in the least interesting part of visiting a theme park: waiting in line for a ride. However, huge investments have been made to create engaging and sometimes interactive experiences during the queue to make the experience feel less painful. Before a Harry Potter-themed ride, visitors roam an immaculate set modeled after Hogwarts, the fictional wizarding school, and the immersion makes the time seem to go by faster.

Theme parks also post estimated wait times so that visitors feel a sense of control about whether or not to join the queue—and these times are padded so that guests feel delight at "saving" time. Isolating wait time provides some information, but if you have set up a standard and even sticking to it leads to unpredictable outcomes, then you must realize that you need more information to guide your decisions.

How do you learn about these expectations? You can look at user-behavior data, such as when customers drop off, and try to figure out a trend. Or you can ask them directly through surveys and interviews. But it's important to think about when is the right time and place to ask them. Asking after a major outage will yield different answers than after a period of calm, and asking them before their issue is resolved is different from asking afterwards.

You will also want to pair up with someone who understands how to craft useful survey questions: for example, you do not want to create leading, ambiguous, or unclear questions, and you want to use a Likert scale. Poorly designed survey questions lead to low quality data, and sometimes people can assume that surveys are the problem, but that's blaming the tool rather than the person wielding it: more than likely it is how surveys are created and conducted that are the problem.

Benefits

We all have limited resources, especially time. When we choose the most meaningful, user-focused SLIs and SLOs, we make the most of those resources. You're prioritizing for the experience your users want and creating the boundaries for services. If something goes down, but it doesn't impact user experience, it's still important, but it isn't necessarily urgent. Just because we can do something doesn't mean we should. We can wake people up in the middle of the night to manage an incident, but are we alerting for the right things? What matters and what doesn't?

There is a broader benefit: the third age of SRE is upon us, and it is one that posits that reliability is cross-functional, something that not just developers and technical project managers need to think of, but also accountants, lawyers, and customer support teams.

Yet this isn't a one-way street. Just as everyone should have a reliability mindset, we must remember why reliability matters. It's not just done for its own sake (and actually can be costly, a detriment to feature velocity, and cause for burnout) but because customer experience matters and customers demand reliability. Reliability that doesn't include a user-focus is only tackling part of the problem, and when it becomes more developer-focused than customer-focused it becomes about ego. So everyone must have a user-oriented mindset.

Such a shift can be frightening because users can seem subjective, but, unless our only users are machines, that's how it goes. What we can do is approach it differently, with wonder and excitement. How can we delight our users the way theme parks spark joy for visitors? Our favorite example of thinking about users: at the Disney World parks, designers created different floor textures for each land, so that even your feet know when you're moving into a new experience. It may be at a level beyond what we need, but we can afford to walk a few steps in the right direction.

References

[1] https://en.wikipedia.org/wiki/Disney_utilidor_system.

[2] M. Rintoul, "User Experience Is a Feeling," UX Matters, October 2014: https://www.uxmatters.com/mt/archives/2014/10/user-experience-is-a-feeling.php.