# Interrupt Reduction Projects

BETSY BEYER, JOHN TOBIN, AND LIZ FONG-JONES

Betsy Beyer is a Technical Writer for Google Site Reliability Engineering in NYC. She has previously provided documentation for Google Data Center and Hardware Operations teams. Before moving to New York, Betsy was a Lecturer in technical writing at Stanford University. She holds degrees from Stanford and Tulane. bbeyer@google.com

John Tobin is a Site Reliability Engineering Manager at Google Dublin. He manages Bigtable, Cloud Bigtable, and a cross-storage SRE automation project, and has worked on several of Google's storage systems. He holds an MSc from Trinity College Dublin, where he also worked before joining Google in 2010. johntobin@google.com

Liz Fong-Jones is a Senior Site Reliability Engineering Manager at Google and manages a team of SREs responsible for Google's storage systems. She lives with her wife, metamour, and two Samoyeds in Brooklyn. In her spare time, she plays classical piano, leads an EVE Online alliance, and advocates for transgender rights. lizf@google.com

Interrupts are a fact of life for any team that's responsible for maintaining a service or software. However, this type of work doesn't have to be a constant drain on your team's bandwidth or resources.

This article begins by describing the landscape of work faced by Site Reliability Engineering (SRE) teams at Google: the types of work we undertake, the logistics of how SRE teams are organized across sites, and the inevitable toil we incur. Within this discussion, we focus on interrupts: how teams initially approached tickets, and why and how we implemented a better strategy. After providing a case study of how the ticket funnel was one such successful initiative, we offer practical advice about mapping what we learned to other organizations.

## Cognitive Flow State and Interrupts

### *Types of Work*

Teams that write and maintain software must decide how to allocate people's time between the main types of work they undertake: planned development, immediate response to outages, and customer requests or lower-urgency production issues.

This article classifies work using the following conventions:

- **On-call/pager response:** Immediate response to outages
- **Tickets and interrupts:** Medium-urgency production issues and customer issues
- **Project work:** Proactive development and systems/network engineering work

In order of most to least urgent, we can make generalizations about how to handle each kind of work.

**On-call/pager response** is critical to the immediate health of the service, and requires a response with an urgency of minutes. Resolving each on-call incident takes between minutes and hours, and our response requires two components: time-sensitive mitigation followed by in-depth investigation and changes to prevent recurrence.

**Tickets and other interrupts** typically have an urgency of days to weeks and usually take between minutes and hours to resolve. These issues frequently prevent the team from achieving reliability goals or are blocking to either internal or external customers. Most teams at Google use a bug or ticket-tracking tool to manage tickets. For simplicity's sake, this article focuses specifically on tickets, the most common form of interrupts handled by our SRE teams.

**Project work** has an urgency ranging from weeks to the long backlog of wishlist ideas every team maintains. This type of work requires multiple days of sustained concentration in order to facilitate cognitive flow state [3]; co-scheduling interrupts or pages with project work will disrupt even the most diligent engineer's focus and prevent them from making meaningful progress.

## Toil and Operational Load

Google categorizes pager response and tickets/interrupts as ***toil***, or reactive work. For a more in-depth discussion of toil, and why and how we seek to minimize it, see Chapter 5 of *Site Reliability Engineering*, "Eliminating Toil" [1], and the follow-up article in the Fall 2016 issue of *;login:*, "Invent More, Toil Less" [2]. Although dealing with toil can provide insight into which properties of the system ought to be improved in the long term, toil itself does not directly provide long-lasting value to a team or service. In a best-case scenario, toil merely allows a team to run in place; in a worst-case scenario, toil consumes enough engineering effort that a service eventually deteriorates. We cap toil at a maximum of 50% of a team's total engineering time, with the expectation that most teams will instead spend 60–70% of their time on project work.

In order to improve a service and reduce the human effort required in maintenance over time, teams must actively work on projects to reduce operational load. As toil decreases, teams can expand their scope to scalably support more services and undertake more interesting project work. Here we focus on one specific category of toil—tickets—and how we successfully reduced their drain on more meaningful project work.

## Context: Google's SRE Team Setup

Most Google SRE teams are spread across two continents, with six to eight people in each of two sites, which together form one team responsible for a given set of services. We assign a primary and secondary on-call in each site, with each site handling 12 hours per day. The primary on-call typically handles most pages. Although your context will be different from ours, the principles we articulate in this article should translate to your organization.

## Initial Approaches to Tickets

### *The Naïve Approach*

Originally, many teams at Google approached tickets by assigning a primary on-call to handle pager duty, while round-robin assigning tickets across the team. This setup frequently led to undesirable outcomes, as engineers couldn't successfully undertake project work and ticket duty simultaneously. Handling random interruptions from tickets prevented engineers from entering a cognitive flow state, so they were unable to achieve meaningful traction on project work. On the other hand, engineers working heads-down on a project missed ticket response expectations because they weren't actively checking for tickets.

Some teams moved in the direction of centralization by assigning tickets to the expert with specialized knowledge or recent experience with a given component. However, this strategy resulted in uneven load and still disrupted people's attention,

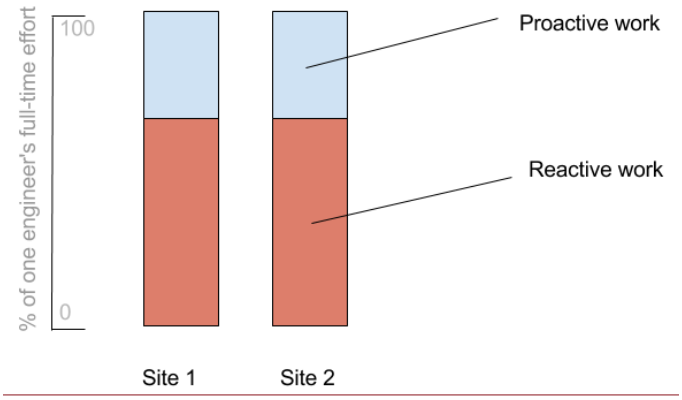


**Figure 1:** Splitting tickets between two people: the optimistic/naïve scenario

making project delivery unpredictable. Delegating the less intellectually interesting work to a team's newest, least experienced team members served only to burn out those team members. We clearly needed a way to dig out of this detrimentally ticket-driven workflow.

### *Centralizing Tickets*

As discussed in the chapter "Dealing with Interrupts" in *Site Reliability Engineering* [1], spreading ticket load across an entire team causes context switches that impact valuable flow time. Once we articulated the need to preserve cognitive flow state [3], a better strategy became clear: we needed to staff a dedicated ticket rotation.

Most SRE teams naïvely implemented this strategy by tasking the secondary on-caller at each site with a somewhat vague and meandering directive:

- Work on tickets until the queue is empty, filing bugs for small improvements as you see ways to improve how specific tickets are handled, or to eliminate them entirely.
- See if you can find commonalities in the tickets you just solved, and do some proactive project-like work to prevent future tickets.

This strategy did at least acknowledge that proactive work is essential to keeping toil manageable as a service increases in size. However, it proved suboptimal: we were resolving tickets but not making small improvements.

In large part, inefficiencies resulted because overall ticket load doesn't necessarily come in whole-person increments. For instance, if your team fields enough tickets to occupy 1.2 people globally per week, you might decide to split the load between two people. While this split would ideally result in the work distribution shown in Figure 1, in actuality, the scenario shown in Figure 2 is much more likely.

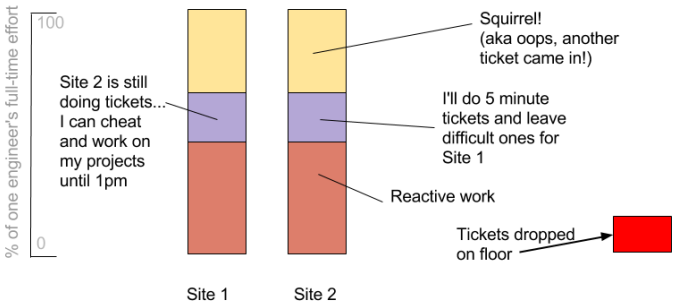


**Figure 2:** Splitting tickets between two people: the actual scenario

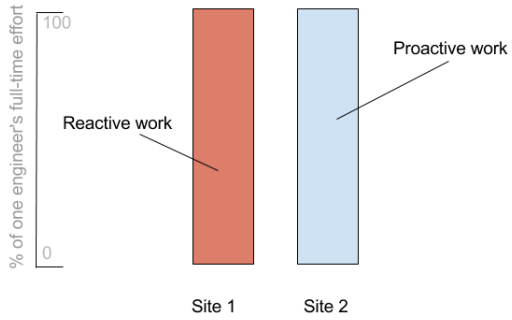


**Figure 3:** Work division through smarter interrupt handling

This setup is far from optimal in terms of cognitive flow, and time zone differences can lead team members to "cheat" on tickets—all too often, engineers at a site only start to work on tickets when coworkers at their partner site go home. At this point, it's tempting to leave frustrating tickets to SREs at your partner site, because you feel less personally connected to those teammates.

For the reasons described above, the ticket handling approach taken by Bigtable SRE (the team John and Liz manage) wasn't working well for a number of years.

## A Better Alternative

We realized that while we initially focused on centralization and fairness/symmetry as a goal, we instead should have focused on maximizing cognitive flow state as a goal in and of itself. Accordingly, we readjusted our goal. We still tasked team members with identifying and solving commonalities in reactive/interrupt-driven work. However, we now explicitly allocated this job, which we'll refer to as "interrupt reduction project on duty," as a separate role from ticket work.

Why is this approach more effective? It hits the sweet spot of undertaking small to medium-sized projects to reduce operational load—projects that require more than 30 minutes of attention, but are too small to account for on a quarterly planning cycle. In our experience, we've identified many such projects that can be completed in less than a week. Assigning one person to work on interrupt reduction projects gives them enough uninterrupted cognitive flow time to complete those projects. Furthermore, assigning one dedicated person to ticket duty at a time ensures accountability for tickets: because that person is singly responsible for tickets, they can't divert responsibility for unresolved tickets onto the other site, or cherry-pick all the easy tickets.

Since instituting these changes, Bigtable SRE is meeting our ticket response expectations more often, and our incoming ticket volume has decreased, as shown in the graphs in Figure 4 (further explained in the case study).

Our division of ticket duty and interrupt reduction project work now looks like Figure 3.

In this new model, we rotate the ticket duty and interrupt reduction project roles between sites once per quarter. In order to ensure fairness, we rotate people into the ticketeer or interrupt reduction rotation according to on-call rotations (e.g., on a weekly basis).

The cost of handing off this new category of small to medium-sized interrupt reduction projects from shift to shift means that our new approach isn't a substitute for undertaking more substantial projects to reduce operational load. Substantive projects are still important—not only for reducing toil in the long term, but for career growth, as well. As such, they should still be accounted for using existing planning and project management processes. We also don't recommend tasking a single person with 12 one-week projects, as doing so would harvest low-hanging fruit, but at an unfair cost to that person's career growth.

## Implementation Details

As we refine the details of our approach to interrupt reduction projects, we've found that the following tactics work well for us.

### *Project Ideas*

Project ideas for the interrupt reduction project on duty come from two main sources:

- Current/past ticket handlers who file annoyances into a bug hotlist as they resolve tickets
- Technical Leads (TLs) who have a high-level view of the service

### *Project Assignment and Handoff*

There are generally many more project ideas than engineering time to implement them, so the TL or someone with an overall view of the service should sort the project list by impact. In the interest of preserving autonomy among team members, we don't suggest assigning projects. Instead, let people choose from the

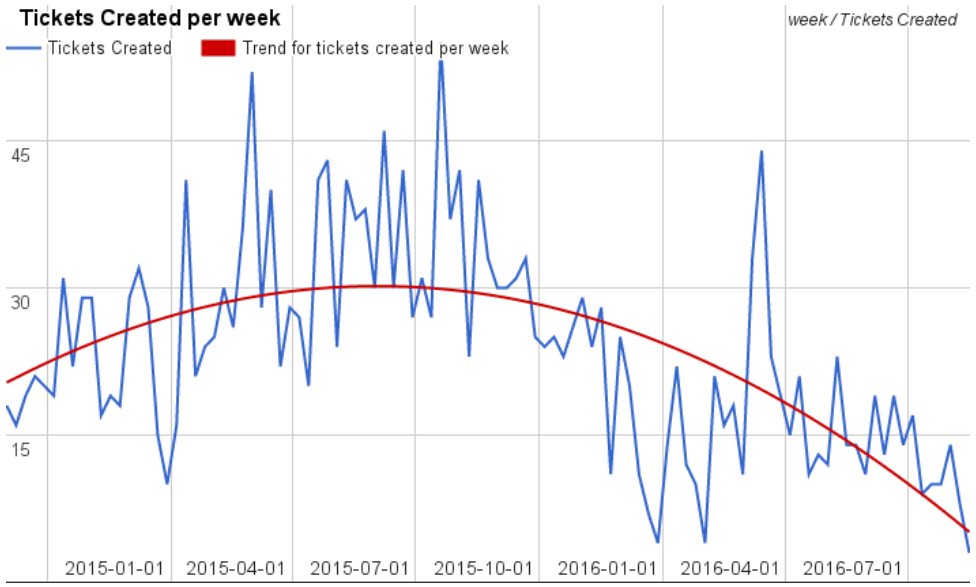


**Figure 4:** Tickets created per week

top 10 projects in the list. If there is any bleed over from last week's project, the interrupt reduction project on duty should finish that project first.

### *Handling Excess Ticket Load*

If your team has more tickets than one person can handle, you have two options for dealing with the excess load:

- Task the interrupt reduction project on duty with tickets for one day per week.
- Decide to relax ticket response expectations for a period of time, until the work pays off.

## Overall Effects

### *Measurable Effects*

We've found that most of the time, one dedicated ticket handler can resolve all tickets, which frees up one person's time for interrupt reduction projects. This is one of the most significant results of our process change, as it proves that some people weren't properly focused on tickets during their ticket duty shifts.

We've reduced overall ticket volume, as shown in the case study that follows (see Figure 4). As a result, we're able to resolve the smaller number of incoming tickets more quickly, although velocity gains are somewhat countered by the increasing difficulty of the tickets we receive—by providing our customers with a better service, we've increased demand for help with complex problems like improving performance.

We complete approximately three of these small strategic interrupt reduction projects every four weeks.

Nooglers (new Googlers) training on the service spend less time on boring or repetitive tickets and more time on interesting tickets that actually create learning opportunities.

### *Less Measurable Effects*

People complain less when they're doing tickets. It feels like we're spending more time on difficult and rewarding tickets and less time on simple or repetitive tickets.

Customers are happier about our ticket handling. Faster ticket resolution helps build goodwill with our customers.

The tension between sites about the effort put into tickets has disappeared, which has led to better overall cross-site relationships.

## Example Case Study: Ticket Funnel

In mid-2015, the Bigtable SRE Team was dealing with a high load of customer tickets. The number of tickets opened per week had increased by roughly 50% over the previous year, from 20+ to 30+ (see Figure 4), and we frequently complained that many tickets could easily be handled by redirecting customers to automation or documentation. Building a simple ticket funnel system to guide customers to appropriate automation or documentation was a natural choice for our first interrupt reduction project.

Instead of directly creating a ticket, customers now work through a simple Web interface where they traverse a decision tree. Non-leaf nodes in the tree are represented as a list of questions linking to child nodes, and leaf nodes do one of the following:

- Link to the relevant self-service automation or documentation.
- Provide a form that generates a customer ticket.

By immediately pointing customers to relevant automation or documentation, we both reduced the number of incoming tickets and improved the quality of service for customers, who no longer have to wait for a human to answer easily resolvable questions. By asking for specific information based on the type of problem, we eliminated an unnecessary round trip of requesting information that could have been supplied up front. Now that this infrastructure is in place, it's easy to update the decision tree with more questions and leaf nodes as we identify more common requests and issues.

Although we'd been talking about undertaking this project for two years, it ended up only taking about two weeks of work. As such, the ticket funnel is a successful example of work that's enabled by the interrupt reduction projects approach: the project was too large to complete on the spur of the moment, but not large enough or important enough to be a standalone project tracked on a longer timescale.

Once our solution was in place, it took a bit of time to reeducate customers, who fully embraced the ticket funnel once they discovered its utility. While measuring tickets that never got filed is difficult, we do have some data that speaks to the ticket funnel's success:

- Figure 4 shows that the ticket creation rate dropped by roughly half after we implemented the ticket funnel, from 30+ to 15+ per week. While we'll never completely eliminate tickets, the overall trend has most definitely reversed.
- Quarterly customer satisfaction surveys reveal an overall happier customer base.
- Anecdotally, we see far fewer tickets that can be resolved by pointing customers at automation or documentation.

## Applying This Strategy in Your Organization

### *Figuring Out a Strategy*

In order to determine how to best apply a similar strategy at your organization, consider the following series of questions.

**How much time does your team have to work on interrupt reduction projects?**

- How much time do you allocate to tickets each week? e.g., *1 person? 4 people?*
- How much time do you actually spend working on tickets each week? e.g., *1 person? 2.5 people?*

Subtract bullet two from bullet one. If you're left with significantly less than 100% of one person's time, you probably won't be able to make meaningful progress on interrupt reduction projects using the slack from ticket duty. If this is the case, you have two possible solutions:

- Spend less time on large projects.
- Spend less time on tickets.

If you choose the second option, you need to think about implementation:

- Will you stop working on tickets entirely? Or will you postpone working on a class of tickets until the automation to deal with them is in place? How will this course of action affect your customers?
- Do you have an expected service level for ticket response time or resolution speed?

The nature of your customers (e.g., internal vs. external) greatly affects not only the answers to these questions, but the expected timeline and impact of the projects. The level of disruption that customers will accept is proportional to the benefit they can expect, so make sure to clearly communicate the motivations for your actions, expected disruptions, timeline, and expected benefits. Market realities will greatly constrain your tactics when it comes to externally visible products, so work with product management and marketing and sales to determine how to accomplish your goals without severely impacting business.

**Who will work on the interrupt reduction projects?**

Make interrupt reduction projects part of the normal ticket duty rotation, which we assume is fairly scheduled and distributed. If that's not the case, think carefully about who will work on these projects. It's important that the work be seen as valuable by the team. Choose people who are enthusiastic or particularly productive in order to create a good initial impression.

**How will you convince your team to adopt this approach?**

Here are some selling points you may be able to use or adapt:

- Each team member will spend 50% less time on tickets.
- Completing a small interrupt reduction project quickly and seeing immediate impact creates a good deal of satisfaction.
- Interrupt reduction projects will improve the systems your team uses on a daily basis.
- Eventually, your ticket load will decrease. The remaining tickets will be issues that actually merit investigation, and improvements to tooling will make some tickets easier to deal with than they were previously.

**How will you safeguard the time allocated for interrupt reduction projects?**

It's tempting for people to ignore interrupt reduction projects in favor of large projects, especially if those large projects have external commitments or interest. Therefore, creating accountability around interrupt reduction projects is important. You might accomplish this by publishing objectives around these projects, reporting on them regularly, tracking them, or announcing interrupt reduction project velocity in regular reports.

## Suggested Interrupt Reduction Projects

The following generic suggestions for interrupt reduction projects should provide a substantial return on the time invested in them.

### *Identify the Sources of Your Toil*

It may seem obvious, but before you can effectively reduce toil, you need to understand the sources of your toil. Consider adding metadata (e.g., cause, impact, time to fix) to tickets to help determine recurring issues and your biggest time sinks.

### *Improve Your Documentation*

Many engineers are allergic to writing documentation, but documentation is a very low cost way to address customer needs and improve ticket handling. It's much easier to handle a ticket if the process is documented, and documentation is a good first step towards automating a process. If customers can find and use good documentation, they won't need to open a ticket.

Often, the blank page effect is the biggest impediment to writing documentation: someone doesn't know where to start, so they don't start. Provide a standard template for each type of documentation (customer facing, internal procedures, <your type here>) to make getting started easier.

### *Pick the 10 Most Annoying Small Bugs and Fix Them*

Your team should be creating lists of bugs for the rough edges, shortcomings, and difficulties encountered in the course of everyday work—otherwise those problems will never be fixed. Pick the ten most annoying small bugs and fix them. Preferably, choose commonly encountered bugs, as people will notice when they're fixed. Consider choosing bugs related to one or two systems, rather than scattered small improvements, so that progress is significant and noticeable. Seeing improvements encourages team members to file bugs, providing a ready source of interrupt reduction projects.

## Takeaways

If tickets/interrupts are an inevitable part of your team's workload, be thoughtful in formulating a strategy to handle them. If you don't implement some type of strategy to proactively reduce tickets, their volume is likely to spiral out of control and become unsustainable in the medium and long term. It's also important to ensure that handling tickets doesn't constantly disrupt the cognitive flow state of your engineers.

Our recommendations for approaching tickets/interrupts, which have been implemented by multiple storage-related services at Google, include four concrete components:

- Centralize your ticket load, either onto engineers who are already expecting interruptions (e.g., primary or secondary on-call) or to a dedicated ticket duty rotation.
- Track ideas for small interrupt reduction projects that will reduce toil.
- Put a framework in place that reserves time for small (20–30 hours) proactive projects.
- Treat tickets and small proactive interrupt reduction projects as separate rotations, distributed among team members and sites on a regular basis.

### *Acknowledgments*

Many thanks to Vladimir Rusinov, who invested heroic effort in analyzing and fixing the root causes of customer tickets, and to Todd Underwood and Dave O'Connor for their input in writing this article.

### *References*

[1] B. Beyer, C. Jones, J. Petoff, and N. Murphy, *Site Reliability Engineering: How Google Runs Production Systems* (O'Reilly Media, 2016).

[2] B. Beyer, B. Gleason, D. O'Connor, and V. Rau, "Invent More, Toil Less," *;login:*, vol. 41, no. 3 (Fall 2016): https://www.usenix.org/publications/login/fall2016/beyer.

[3] Flow: https://en.wikipedia.org/wiki/Flow_(psychology).

# SRECON®

SREcon is a gathering of engineers who care deeply about site reliability, systems engineering, and working with complex distributed systems at scale. It strives to challenge both those new to the profession as well as those who have been involved in it for decades. The conference has a culture of critical thought, deep technical insights, continuous improvement, and innovation.

## SRECON17 AMERICAS

MARCH 13–14, 2017 • SAN FRANCISCO, CA, USA

www.usenix.org/srecon17americas

**The full program and registration will be available in January 2017.**

## SRECON17 ASIA/AUSTRALIA

MAY 22–24, 2017 • SINGAPORE

www.usenix.org/srecon17asia

## SRECON17 EUROPE/MIDDLE EAST/AFRICA

AUGUST 30–SEPTEMBER 2, 2017 • DUBLIN, IRELAND

www.usenix.org/srecon17europe

usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION