Chapter 5 - Eliminating Toil

Eliminating Toil

Written by Vivek Rau
Edited by Betsy Beyer

> If a human operator needs to touch your system during normal operations, you have a bug. The definition of normal changes as your systems grow.
>
> Carla Geisser, Google SRE

In SRE, we want to spend time on long-term engineering project work instead of operational work. Because the term *operational work* may be misinterpreted, we use a specific word: *toil*.

Toil Defined

Toil is not just "work I don't like to do." It's also not simply equivalent to administrative chores or grungy work. Preferences as to what types of work are satisfying and enjoyable vary from person to person, and some people even enjoy manual, repetitive work. There are also administrative chores that have to get done, but should not be categorized as toil: this is *overhead*. Overhead is often work not directly tied to running a production service, and includes tasks like team meetings, setting and grading goals,[19] snippets,[20] and HR paperwork. Grungy work can sometimes have long-term value, and in that case, it's not toil, either. Cleaning up the entire alerting configuration for your service and removing clutter may be grungy, but it's not toil.

So what *is* toil? Toil is the kind of work tied to running a production service that tends to be manual, repetitive, automatable, tactical, devoid of enduring value, and that scales linearly as a service grows. Not every task deemed toil has all these attributes, but the more closely work matches one or more of the following descriptions, the more likely it is to be toil:

Manual

> This includes work such as manually running a script that automates some task. Running a script may be quicker than manually executing each step in the script, but the *hands-on* time a human spends running that script (not the elapsed time) is still toil time.

Repetitive

> If you're performing a task for the first time ever, or even the second time, this work is not toil. Toil is work you do over and over. If you're solving a novel problem or inventing a new solution, this work is not toil.

Automatable

> If a machine could accomplish the task just as well as a human, or the need for the task could be designed away, that task is toil. If human judgment is essential for the task, there's a good chance it's not toil.[21]

Tactical

> Toil is interrupt-driven and reactive, rather than strategy-driven and proactive. Handling pager alerts is toil. We may never be able to eliminate this type of work completely, but we have to continually work toward minimizing it.

No enduring value

> If your service remains in the same state after you have finished a task, the task was probably toil. If the task produced a permanent improvement in your service, it probably wasn't toil, even if some amount of grunt work—such as digging into legacy code and configurations and straightening them out—was involved.

O(n) with service growth

> If the work involved in a task scales up linearly with service size, traffic volume, or user count, that task is probably toil. An ideally managed and designed service can grow by at least one order of magnitude with zero additional work, other than some one-time efforts to add resources.

Why Less Toil Is Better

Our SRE organization has an advertised goal of keeping operational work (i.e., toil) below 50% of each SRE's time. At least 50% of each SRE's time should be spent on engineering project work that will either reduce future toil or add service features. Feature development typically focuses on improving reliability, performance, or utilization, which often reduces toil as a second-order effect.

We share this 50% goal because toil tends to expand if left unchecked and can quickly fill 100% of everyone's time. The work of reducing toil and scaling up services is the "Engineering" in Site Reliability Engineering. Engineering work is what enables the SRE organization to scale up sublinearly with service size and to manage services more efficiently than either a pure Dev team or a pure Ops team.

Furthermore, when we hire new SREs, we promise them that SRE is not a typical Ops organization, quoting the 50% rule just mentioned. We need to keep that promise by not allowing the SRE organization or any subteam within it to devolve into an Ops team.

Calculating Toil

If we seek to cap the time an SRE spends on toil to 50%, how is that time spent?

There's a floor on the amount of toil any SRE has to handle if they are on-call. A typical SRE has one week of primary on-call and one week of secondary on-call in each cycle (for discussion of primary versus secondary on-call shifts, see Being On-Call). It follows that in a 6-person rotation, at least 2 of every 6 weeks are dedicated to on-call shifts and interrupt handling, which means the lower bound on potential toil is 2/6 = 33% of an SRE's time. In an 8-person rotation, the lower bound is 2/8 = 25%.

Consistent with this data, SREs report that their top source of toil is interrupts (that is, non-urgent service-related messages and emails). The next leading source is on-call (urgent) response, followed by releases and pushes. Even though our release and push processes are usually handled with a fair amount of automation, there's still plenty of room for improvement in this area.

Quarterly surveys of Google's SREs show that the average time spent toiling is about 33%, so we do much better than our overall target of 50%. However, the average doesn't capture outliers: some SREs claim 0% toil (pure development projects with no on-call work) and others claim 80% toil. When individual SREs report excessive toil, it often indicates a need for managers to spread the toil load more evenly across the team and to encourage those SREs to find satisfying engineering projects.

What Qualifies as Engineering?

Engineering work is novel and intrinsically requires human judgment. It produces a permanent improvement in your service, and is guided by a strategy. It is frequently creative and innovative, taking a design-driven approach to solving a problem—the more generalized, the better. Engineering work helps your team or the SRE organization handle a larger service, or more services, with the same level of staffing.

Typical SRE activities fall into the following approximate categories:

Software engineering

: Involves writing or modifying code, in addition to any associated design and documentation work. Examples include writing automation scripts, creating tools or frameworks, adding service features for scalability and reliability, or modifying infrastructure code to make it more robust.

Systems engineering

: Involves configuring production systems, modifying configurations, or documenting systems in a way that produces lasting improvements from a one-time effort. Examples include monitoring setup and updates, load balancing configuration, server configuration, tuning of OS parameters, and load balancer setup. Systems engineering also includes consulting on architecture, design, and productionization for developer teams.

Toil

: Work directly tied to running a service that is repetitive, manual, etc.

Overhead

: Administrative work not tied directly to running a service. Examples include hiring, HR paperwork, team/company meetings, bug queue hygiene, snippets, peer reviews and self-assessments, and training courses.

Every SRE needs to spend at least 50% of their time on engineering work, when averaged over a few quarters or a year. Toil tends to be spiky, so a steady 50% of time spent on engineering may not be realistic for some SRE teams, and they may dip below that target in some quarters. But if the fraction of time spent on projects averages significantly below 50% over the long haul, the affected team needs to step back and figure out what's wrong.

Is Toil Always Bad?

Toil doesn't make everyone unhappy all the time, especially in small amounts. Predictable and repetitive tasks can be quite calming. They produce a sense of accomplishment and quick wins. They can be low-risk and low-stress activities. Some people gravitate toward tasks involving toil and may even enjoy that type of work.

Toil isn't always and invariably bad, and everyone needs to be absolutely clear that some amount of toil is unavoidable in the SRE role, and indeed in almost any engineering role. It's fine in small doses, and if you're happy with those small doses, toil is not a problem. Toil becomes toxic when experienced in large quantities. If you're burdened with too much toil, you should be very concerned and complain loudly. Among the many reasons why too much toil is bad, consider the following:

Career stagnation

: Your career progress will slow down or grind to a halt if you spend too little time on projects.

Google rewards grungy work when it's inevitable and has a big positive impact, but you can't make a career out of grunge.

Low morale

People have different limits for how much toil they can tolerate, but everyone has a limit. Too much toil leads to burnout, boredom, and discontent.

Additionally, spending too much time on toil at the expense of time spent engineering hurts an SRE organization in the following ways:

Creates confusion

We work hard to ensure that everyone who works in or with the SRE organization understands that we are an engineering organization. Individuals or teams within SRE that engage in too much toil undermine the clarity of that communication and confuse people about our role.

Slows progress

Excessive toil makes a team less productive. A product's feature velocity will slow if the SRE team is too busy with manual work and firefighting to roll out new features promptly.

Sets precedent

If you're too willing to take on toil, your Dev counterparts will have incentives to load you down with even more toil, sometimes shifting operational tasks that should rightfully be performed by Devs to SRE. Other teams may also start expecting SREs to take on such work, which is bad for obvious reasons.

Promotes attrition

Even if you're not personally unhappy with toil, your current or future teammates might like it much less. If you build too much toil into your team's procedures, you motivate the team's best engineers to start looking elsewhere for a more rewarding job.

Causes breach of faith

New hires or transfers who joined SRE with the promise of project work will feel cheated, which is bad for morale.

Conclusion

If we all commit to eliminate a bit of toil each week with some good engineering, we'll steadily clean up our services, and we can shift our collective efforts to engineering for scale, architecting the next generation of services, and building cross-SRE toolchains. Let's invent more, and toil less.

[19] We use the Objectives and Key Results system, pioneered by Andy Grove at Intel; see [Kla12].

[20] Googlers record short free-form summaries, or "snippets," of what we've worked on each week.

[21] We have to be careful about saying a task is "not toil because it needs human judgment." We need to think carefully about whether the nature of the task intrinsically requires human judgment and cannot be addressed by better design. For example, one could build (and some have built) a service that alerts its SREs several times a day, where each alert requires a complex response involving plenty of human

judgment. Such a service is poorly designed, with unnecessary complexity. The system needs to be simplified and rebuilt to either eliminate the underlying failure conditions or deal with these conditions automatically. Until the redesign and reimplementation are finished, and the improved service is rolled out, the work of applying human judgment to respond to each alert is definitely toil.

previous

Chapter 4 - Service Level Objectives

next

Chapter 6 - Monitoring Distributed Systems