Zero-bug policy success

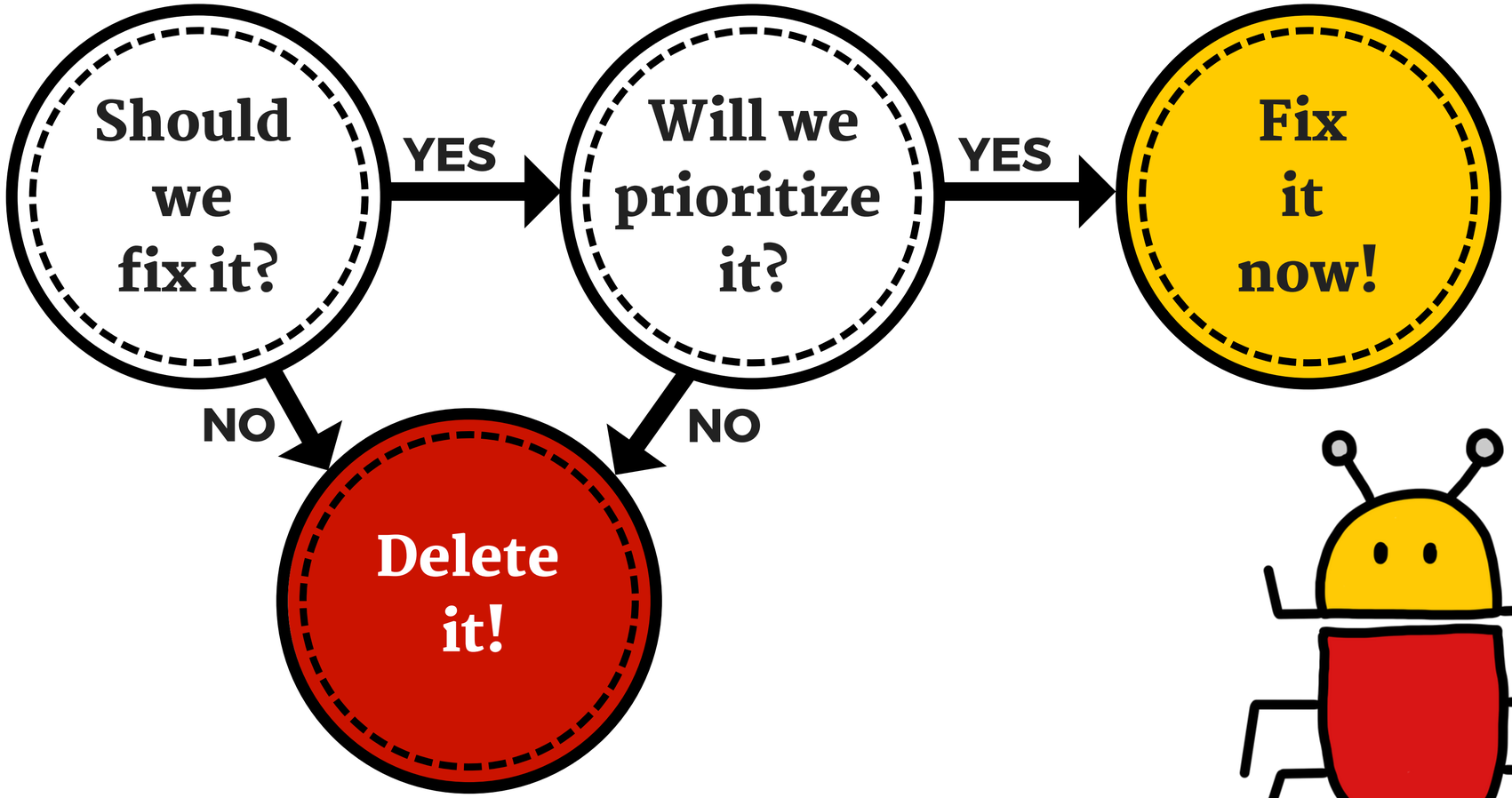
Peter Hilton

@hilton.org.uk

http://hilton.org.uk/

FIX IT NOW OR DELETE IT

THE DEFINITIVE BUG MANAGEMENT SYSTEM

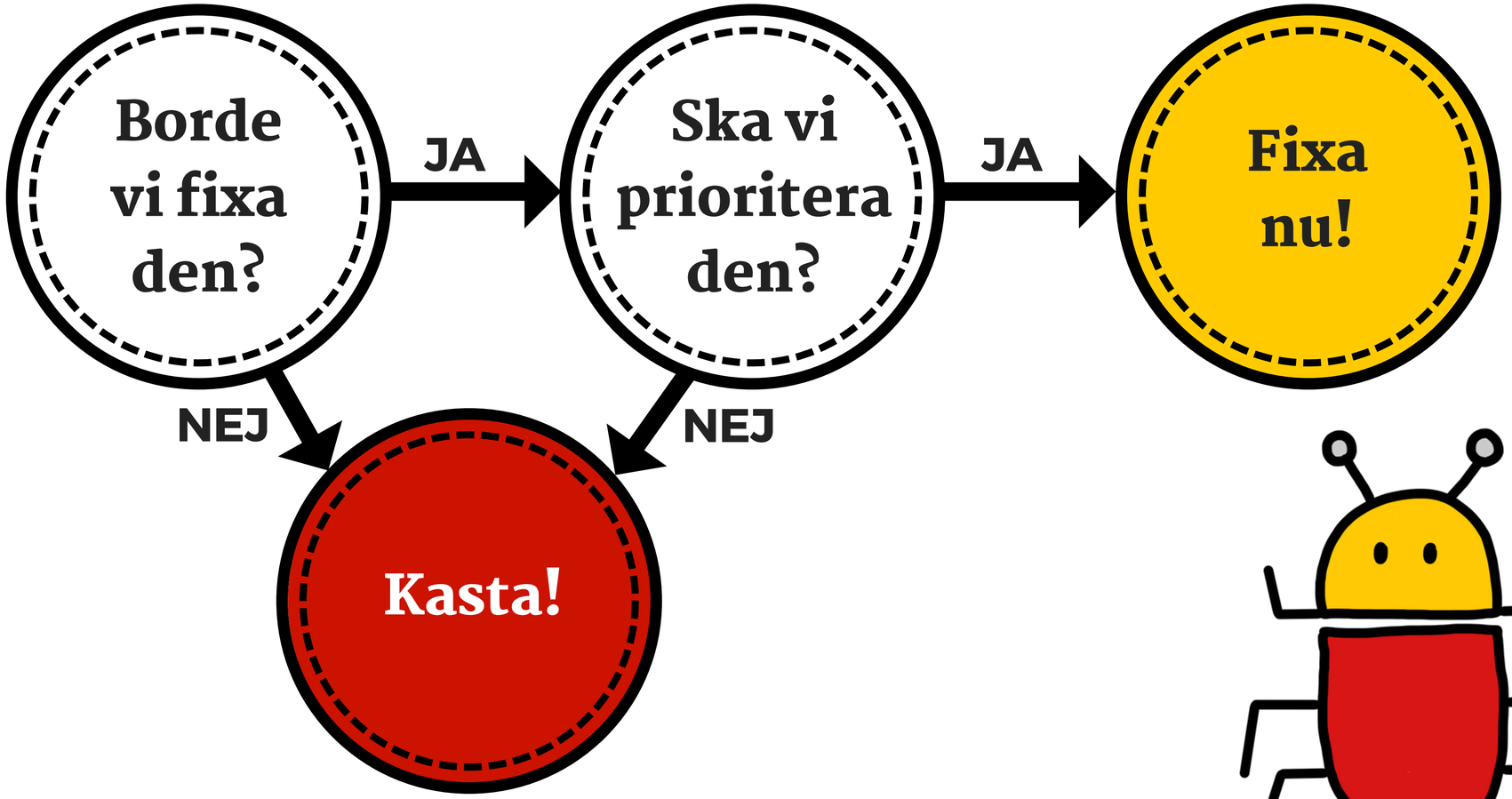


FIXITNOWORDELETEIT.COM

BY YASSAL SUNDMAN

FIXA NU ELLER KASTA

DET ULTIMATA BUGGHANTERINGSSYSTEMET



FIXITNOWORDELETEIT.COM

AV YASSAL SUNDMAN

Policies, targets & systems

The notion of a zero-bug policy is sometimes controversial,
but zero bugs is a target, not a policy.

‘Zero-bug policy’ is an inaccurate name for
a bug management system for achieving a target.

(naming things is hard 🤷‍♂️)

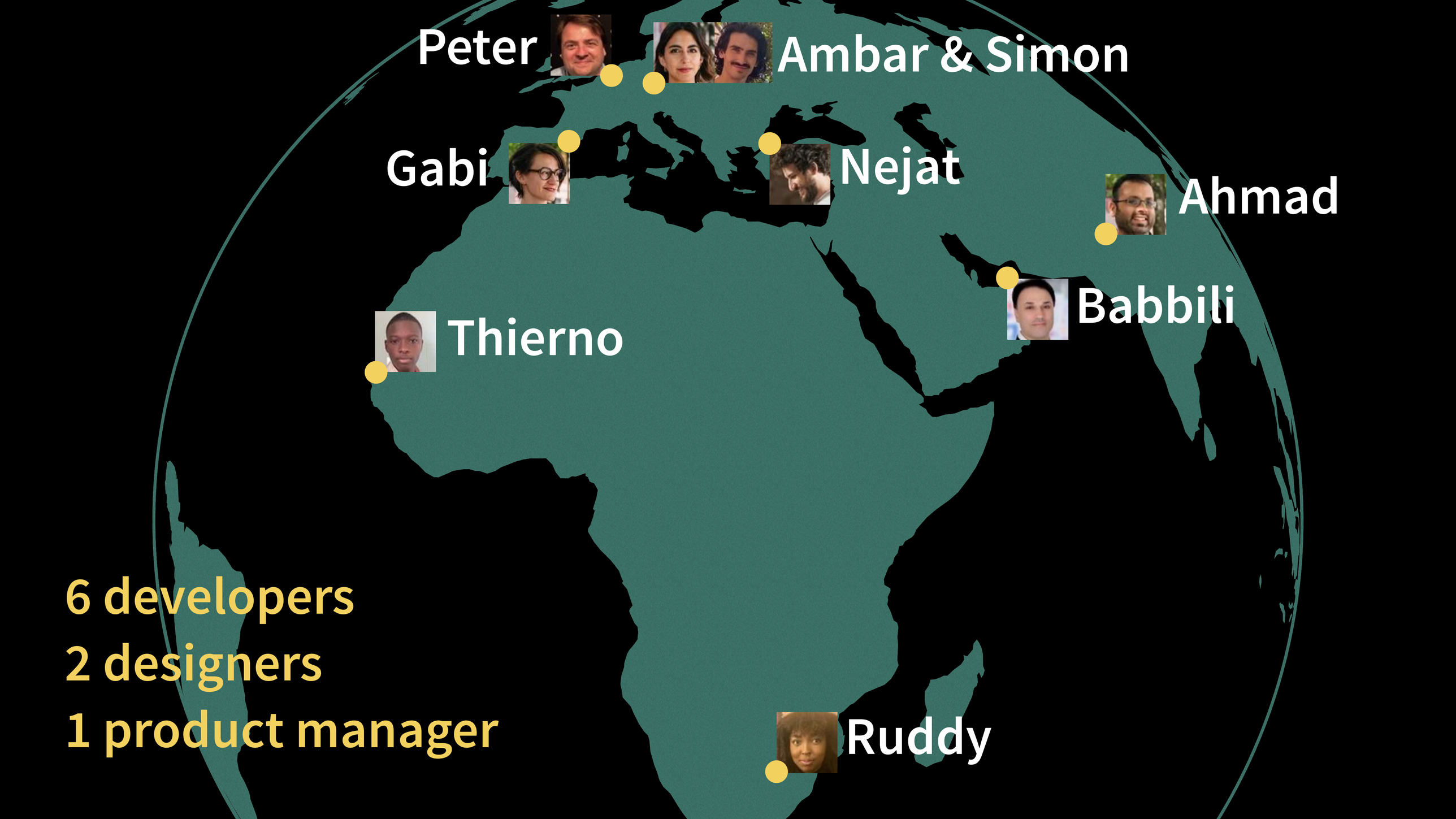
Context - who, what, where

Context matters

Most of what you hear and read about developing software supposedly applies to… the whole industry 🙄

But as every consultant (and product manager) knows:
it depends

This is why we need experience reports


Peter
Ambar & Simon
Gabi
Nejat
Ahmad
Babbili
Thierno
Ruddy
6 developers
2 designers
1 product manager

Slik fungerer det

Bestill produkter

Bestill produktene du trenger, og få påfyll levert hjem i beholdere som brukes igjen og igjen.

Fyll på

Produktene heller du over på mindre dispensere, gjerne noen du har fra før. På(fyll)-beholderne kan du lagre i en skuff eller i et skap og ta frem når du trenger å fylle på igjen.

Les mer om dispensere

Scan og bestill

Når beholderen er tom, kan du enkelt bestille nytt På(fyll) ved å scanne QR-koden på beholderen, eller gå inn på påfyll.no.

Få påfyll hjem

Du får beskjed straks bestillingen er på vei. De tomme beholderne putter du i gjenbruksesken de kom i og setter den utenfor døra. Vi henter dem, vasker og fyller på, og sender ut til nye På(fyll)-kunder.




På(fyll) er en ny og sirkulær tjeneste som sørger for at du alltid har påfyll av produktene du bruker tilgjengelig.

Finn ut hvordan På(fyll) fungerer

Bli kjent med På(fyll)

Så enkelt får du hverdagsproduktene levert hjem til deg

Kundehistorie

Slik blir du "plastsmart" med På(fyll)

Sirkulærøkonomi

Enjoy free shipping on orders over 295 NOK

På(fyll) is a new circular service that ensures you always have the products you need.

Find out how På(fyll) works

How it works

How it works

Order exactly what you need from our online platform, and we'll deliver freshly filled containers directly to your doorstep.

Fill dispensers

If you have them, fill up any dispensers with product from your containers, and then you're ready to go.

Read more

Scan to reorder

Whenever you're running low, scan the QR code on your container or go to pafyll.com to order a refill.

Receive your refill

We'll deliver another freshly filled container straight away. Leave your empty ones outside, and we'll collect them to be washed, refilled, and used again and again.




Learn more about På(fyll)

Delivery

About På(Fyll)

The På(fyll) container

Inspiration

How it works

How it works

Order exactly what you need from our online platform, and we'll deliver freshly filled containers directly to your doorstep.

Fill dispensers

If you have them, fill up any dispensers with product from your containers, and then you're ready to go.

Read more

Scan to reorder

Whenever you're running low, scan the QR code on your container or go to pafyll.com to order a refill.

Receive your refill

We'll deliver another freshly filled container straight away. Leave your empty ones outside, and we'll collect them to be washed, refilled, and used again and again.

Context

Team

Fully-remote

9 people + CTO

Software

Consumer retail

Web shop

Supply chain

CRM + ERP

Circular service

Company

Start-up

20 employees

Mission-driven

(sustainability)

Getting started

YOUR HOST

AUGUST 9, 2000 *by* JOEL SPOLSKY

The Joel Test: 12 Steps to Better Code

TOP 10, ROCK STAR DEVELOPER, NEWS

Have you ever heard of SEMA? It's a fairly esoteric system for measuring how good a software team is. No, *wait! Don't follow that link!* It will take you about six years just to *understand* that stuff. So I've come up with my own, highly irresponsible, sloppy test to rate the quality of a software team. The great part about it is that it takes about 3 minutes. With all the time you save, you can go to medical school.

https://www.joelonsoftware.com/2000/08/09/the-joel-test-12-steps-to-better-code/

The Joel Test: 12 Steps to Better Code (2000)

1. Do you use source control?
2. Can you make a build in one step?
3. Do you make daily builds?
4. Do you have a bug database?
5. Do you fix bugs before writing new code? 👀
6. Do you have an up-to-date schedule?
7. Do you have a spec?
8. Do programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?

Implement a zero-bug policy

How to manage bugs during software product development • 2021-01-05

Katie Blackmore

Product development teams typically don't use their own software enough to understand their customers' experience, and underestimate how much their customers hate bugs. Meanwhile, many software development teams fix some bugs, but not others, and end up tracking hundreds of open bugs. This wastes time, and does not deliver value.

The Joel Test addressed this twenty years ago with this question:

> 5. Do you fix bugs before writing new code?

Spolsky explains that a team with a large number of open bugs has a worse problem than the time wasted managing that list: unknown liability for future work. Software development schedules have no meaning with open bugs. Fixing bugs first creates predictability.

Alternatively, save yourself some time by skipping both Spolsky's article and this one, and immediately implement Yassal Sundman's bug management system: Fix It Now or Delete It!

Adopt a zero-bug policy

Fixing bugs before you write new code means adopting a zero-bug policy:

1. When you discover a bug, fix it before other product development work, by default.
2. If fixing the bug delivers little value, discard the bug report instead.
3. Don't keep low-priority bugs unfixed for 'later'.

Anything else encourages releasing unfinished work and success theatre. Keeping bugs for 'later' also leads to double-handling, such as (repeatedly) reprioritising to *even lower*, rescheduling to *even later*, and reproducing buggy behaviour.

A zero-bug policy has an unexpected consequence at the start: you have to discard all bug reports that you won't resolve before resuming other work. If this sounds extreme, consider that it won't result in fixing fewer bugs.

Decide what to fix

Introducing a zero-bug policy does not mean fixing every bug: product managers have to decide what to ignore. Otherwise, you'll incur the opportunity cost of fixing issues that don't affect customers. Meanwhile, some bug reports describe improvements to existing functionality, or missing features. Further reading:

- Stop Managing Bugs, Start Focusing on Quality! and The Definitive Bug Management System, by Yassal Sundman, advocate deleting bug reports.
- Zero-Bug Software Development, by Sam Hatoum, discusses reclassifying bugs as *improvements* or *new features*.

Choosing what to classify as a bug shouldn't require discussion. If product managers and developers don't agree on what to call a bug, you probably need to address other problems.

Continue discovering bugs

People who expect a zero bug policy to mean software that never has bugs clearly don't understand software development. Don't get upset by the *zero-bug* name, or confuse it with zero defects in industrial production; consider *zero bugs* as an achievable objective. Call it the *zero bugs objective* if you like.

You will continue to discover bugs, but you can have zero *known bugs* most of the time. If you don't regularly achieve bug inbox zero, you have other problems, such as ineffective testing or unmaintainable code that slows down bug fixing.

Increase development speed

Zero-bug policies don't only improve software quality. Avoiding managing bugs and having meetings about unpredictable development schedules gives you more time for product development.

Further reading: How a zero-bug policy doubled our development velocity, by Jan Peratt, explains how 'fixing bugs continuously increases long-term velocity'.

From idea to adoption (2024)

January	Team retro idea to ‘Adopt a zero-bug policy’.
	Peter’s blog post shared on Slack.
February	Zero-bug policy team discussion.
March	My first day, discussed it again in the team retro.
April	Marketing campaign leads to many bug reports.

Bugs hurt 🩸

NIAID

J. B. Rainsberger
@jbrains.ca

Why refactor? To reduce volatility in the marginal cost of features.

We can see this as a socially acceptable way of saying "to express love", especially in a job context.

What would happen if we embraced this idea? Why don't you embrace this idea now? What obstacles stand in your path?

November 12, 2024 at 7:42 AM Everybody can reply

6 reposts **1** quote **6** likes

7 6

The edge case discussion trap: *what if… ?*

1. … we have an existing bug backlog → fix them all!
2. … we didn’t fix the bug quickly → evaluate why not!
3. … the bug doesn’t affect customers → fix it anyway!
4. … the fix requires even more time → re-evaluate!
5. … we have multiple open bugs → prioritise!
6. … there’s a critical bug → interrupt other work!

Vision

 Team happiness 

Goals

🎯 Predictable effort to support & maintain a stable system.

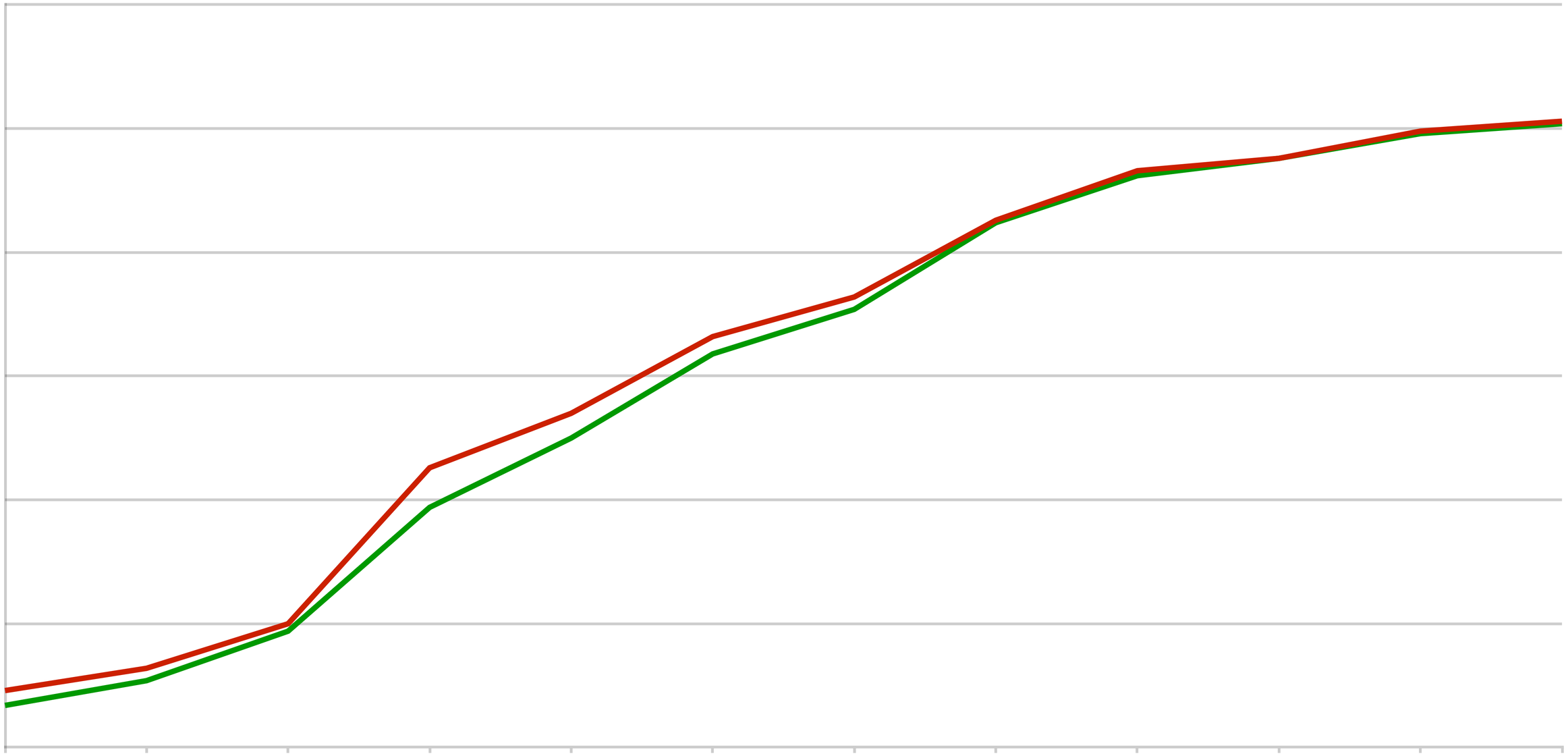
🎯 Stop wasting time discussing and prioritising open bugs.

🎯 Meaningful product feedback from feature adoption data (because customers don’t use broken features).

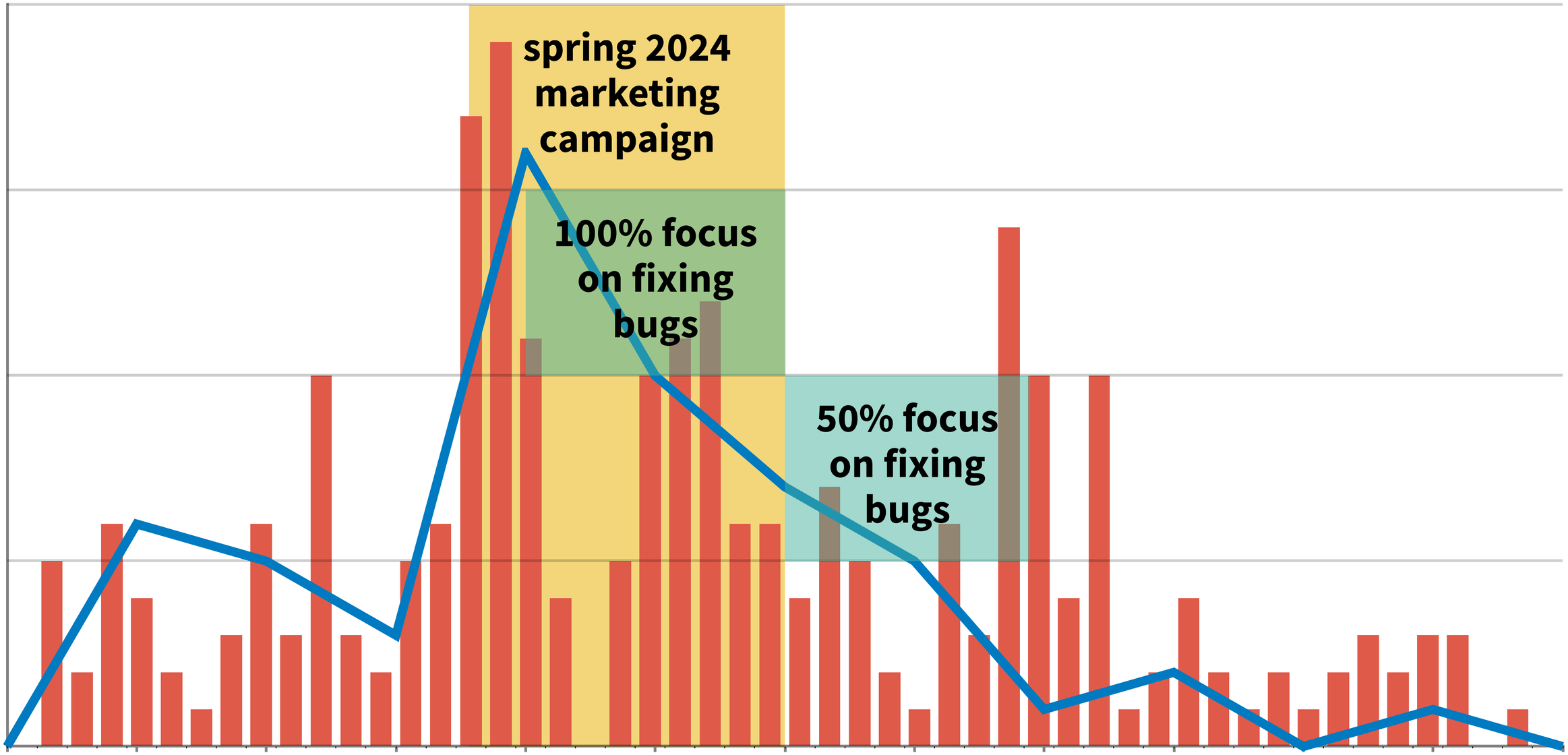
Safe to try

The bug workload

Bug workload 2024



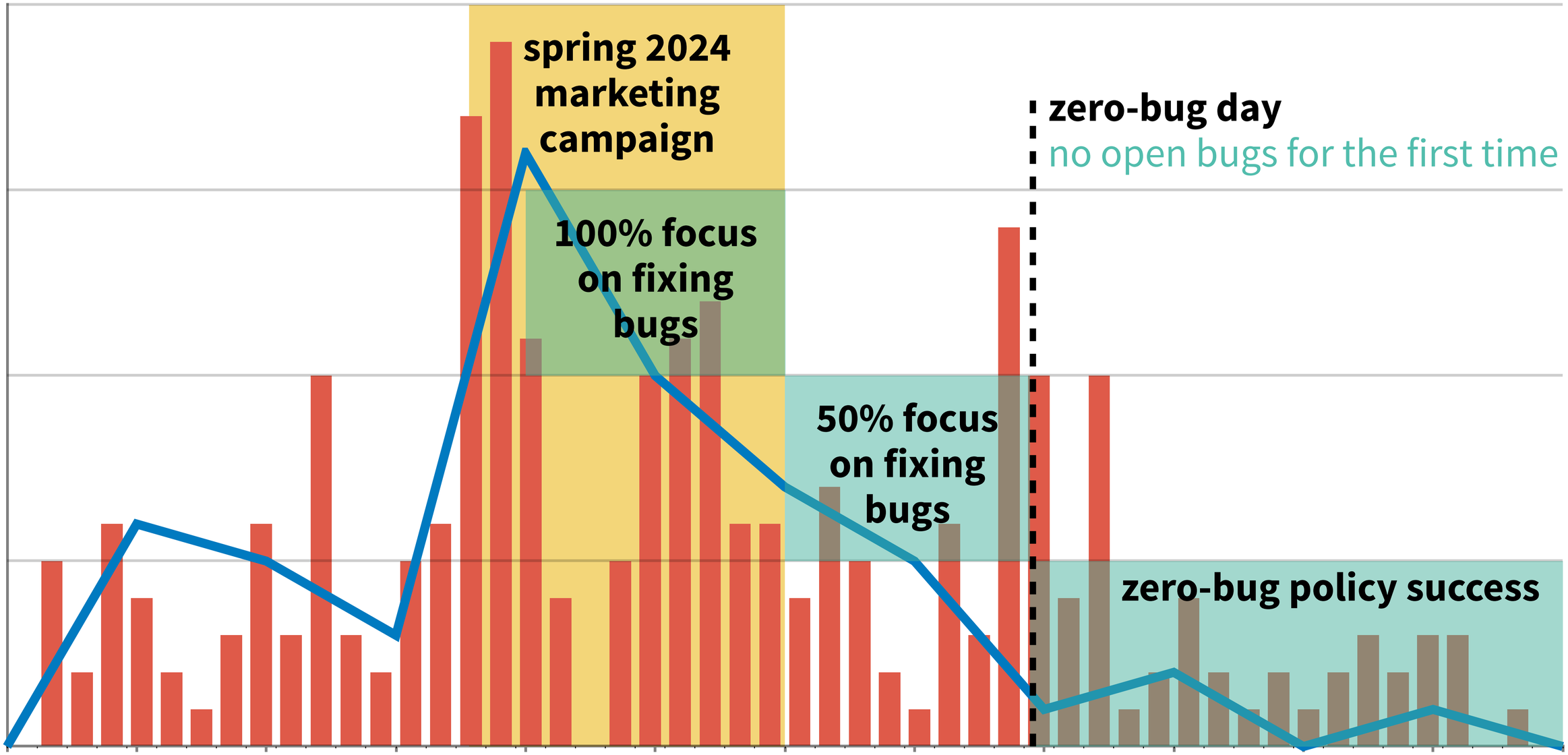
Bug workload 2024



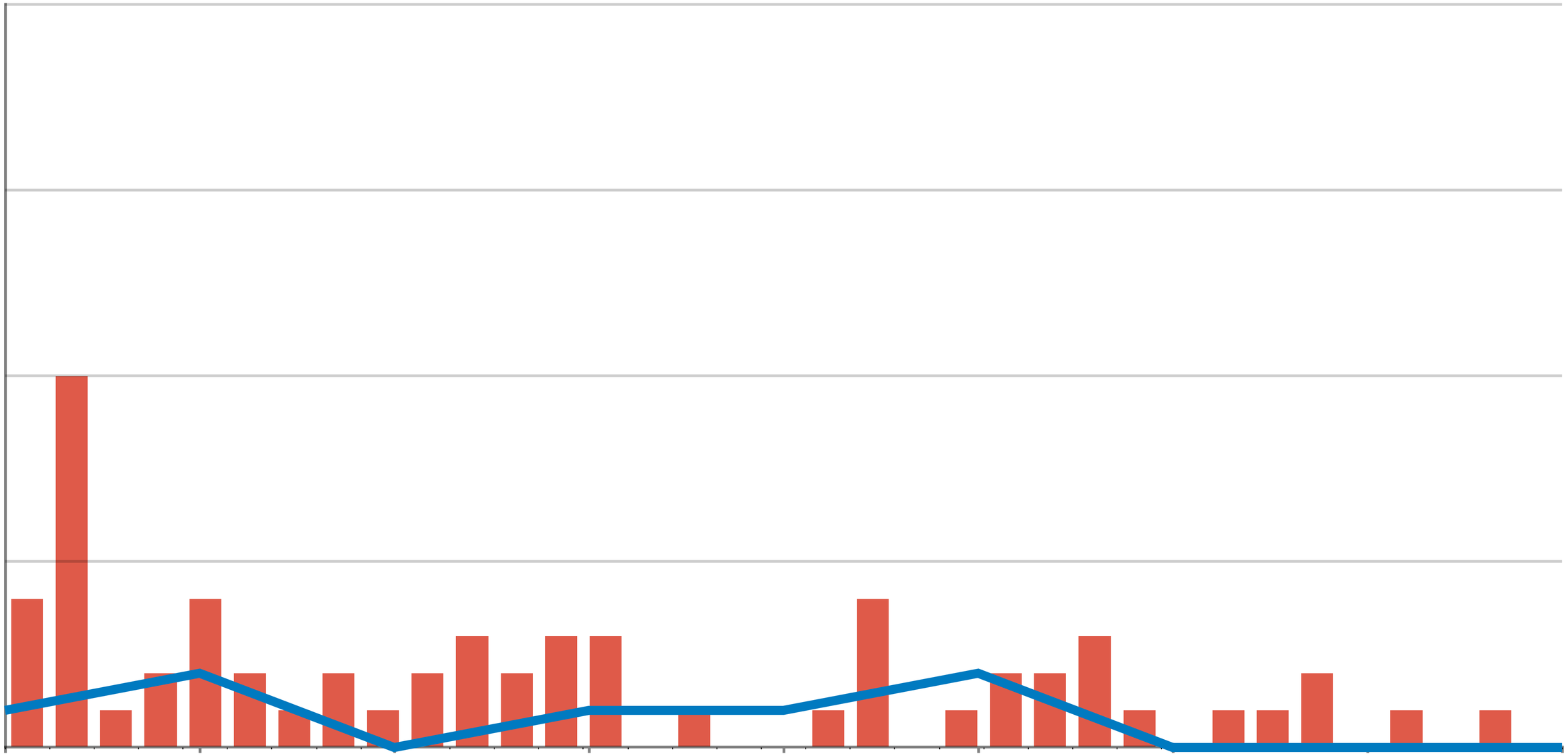
‘Zero bugs policy pissed me off this week’

– team retro, August.

Bug workload 2024



September 2024 – May 2025



Few critical bugs

 161 bugs fixed from May 2024 – April 2025

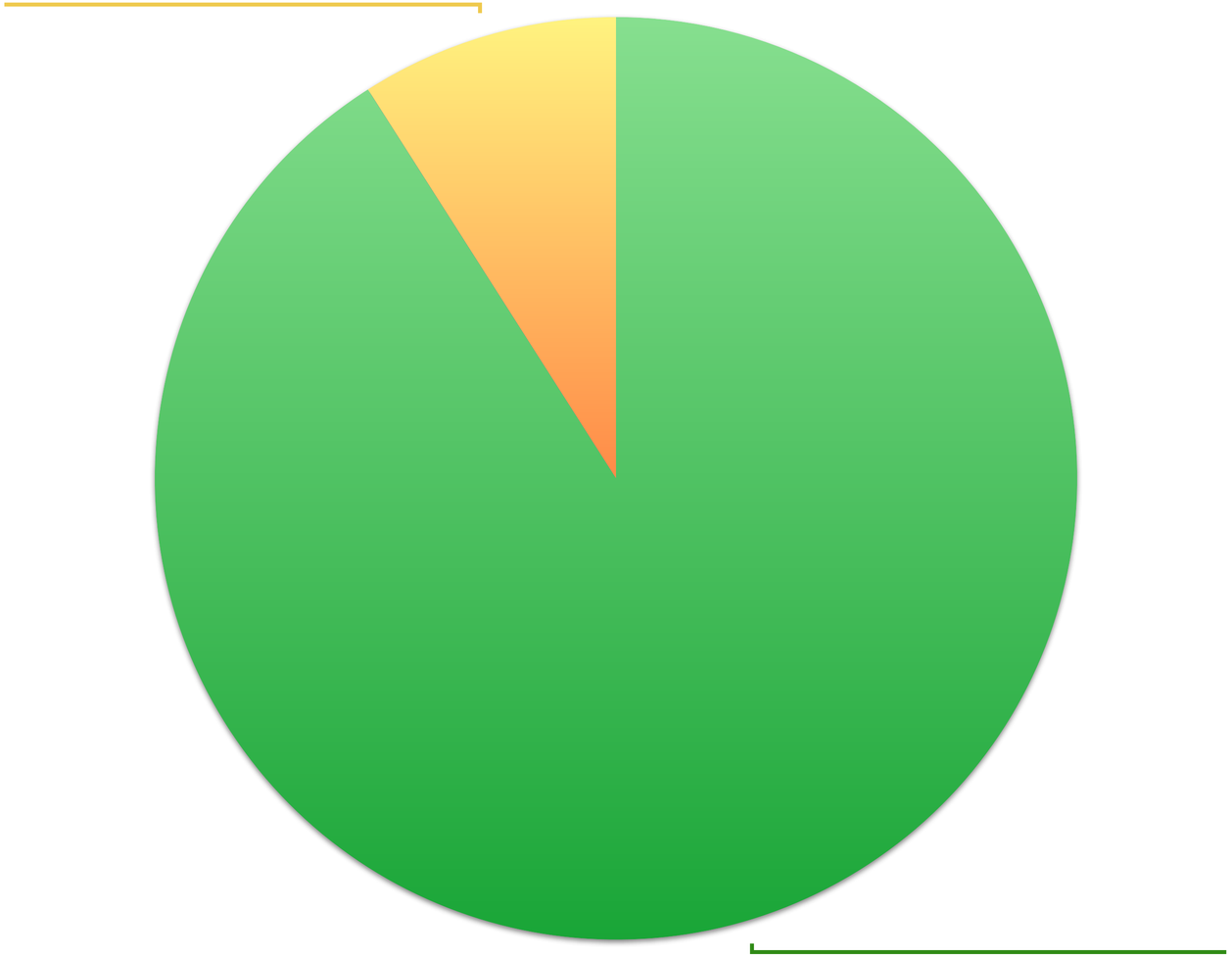
 3 were critical bugs

Only drop work in progress for critical bugs

(normal bug reports are just next in the queue)

Team programming for critical bugs, because FOMO

🐛 Reported bugs May 2024 – April 2025

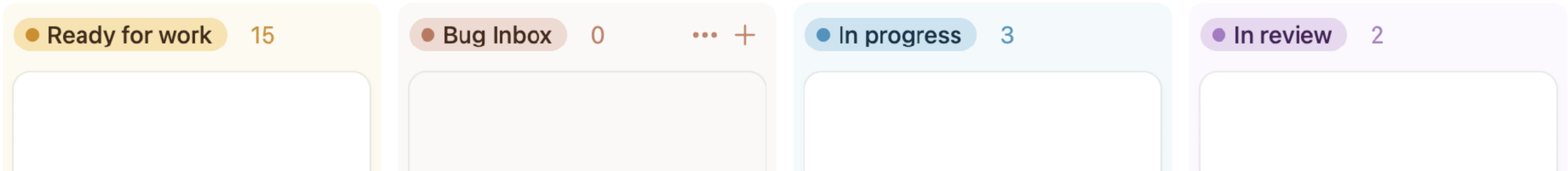


MAY 2025

Mon	Tue	Wed	Thu	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Processes & tools

1. Slack #bugs channel → bug reports, updates 👀🎟️✅❌
2. Weekly @product-support group rota for first responder
3. Notion board (Kanban) view → bug reports 🐛🐞🪲
4. Separate Bug inbox board column (hidden when empty)
5. ‘Air-gapped’ separate product feedback database



Using a bug tracker

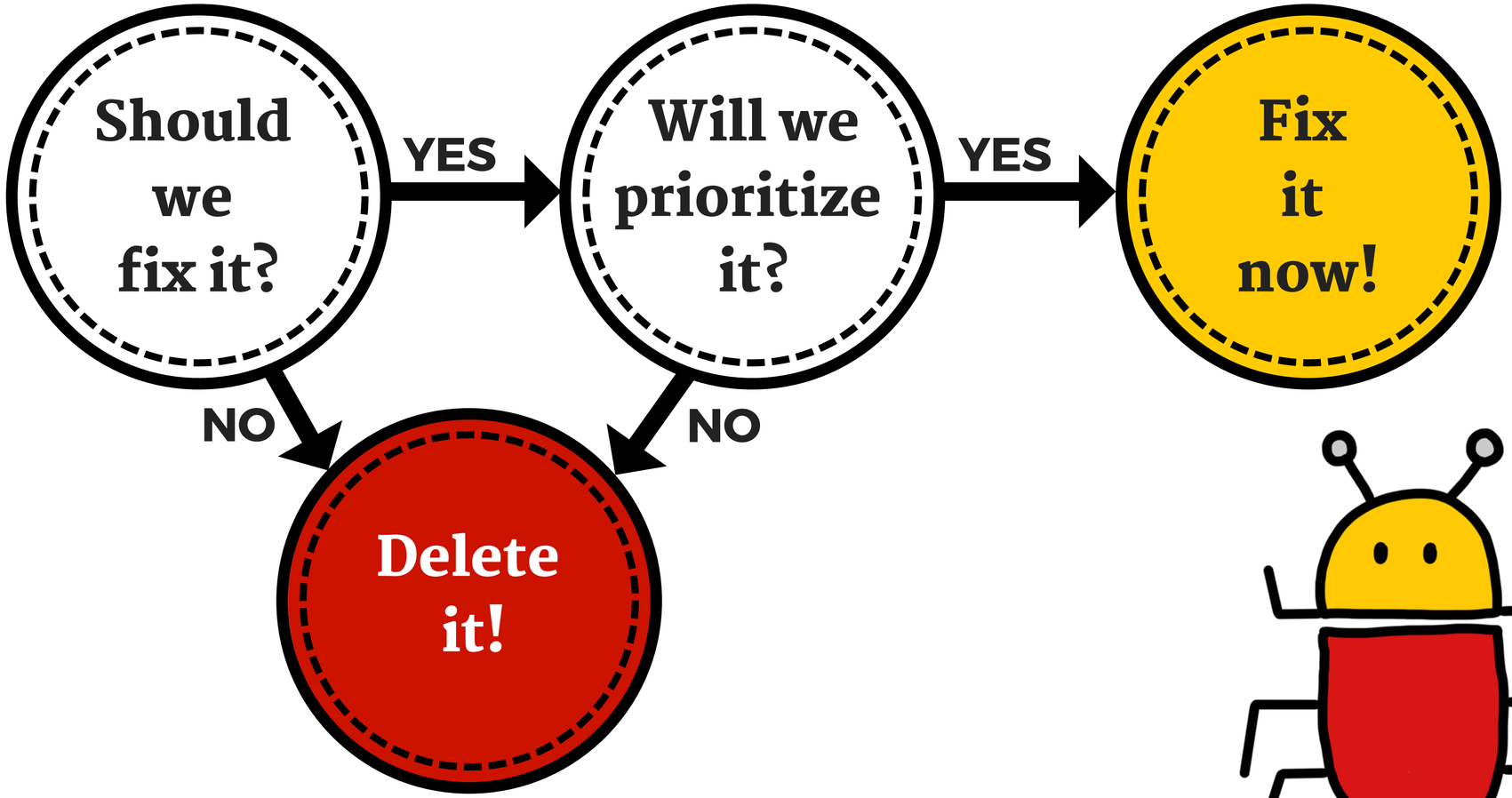
Not tracking bugs

Lessons learned

When you only have one bug at a time, you can deal with them case-by-case.

FIX IT NOW OR DELETE IT

THE DEFINITIVE BUG MANAGEMENT SYSTEM



FIXITNOWORDELETEIT.COM

BY YASSAL SUNDMAN

Transition

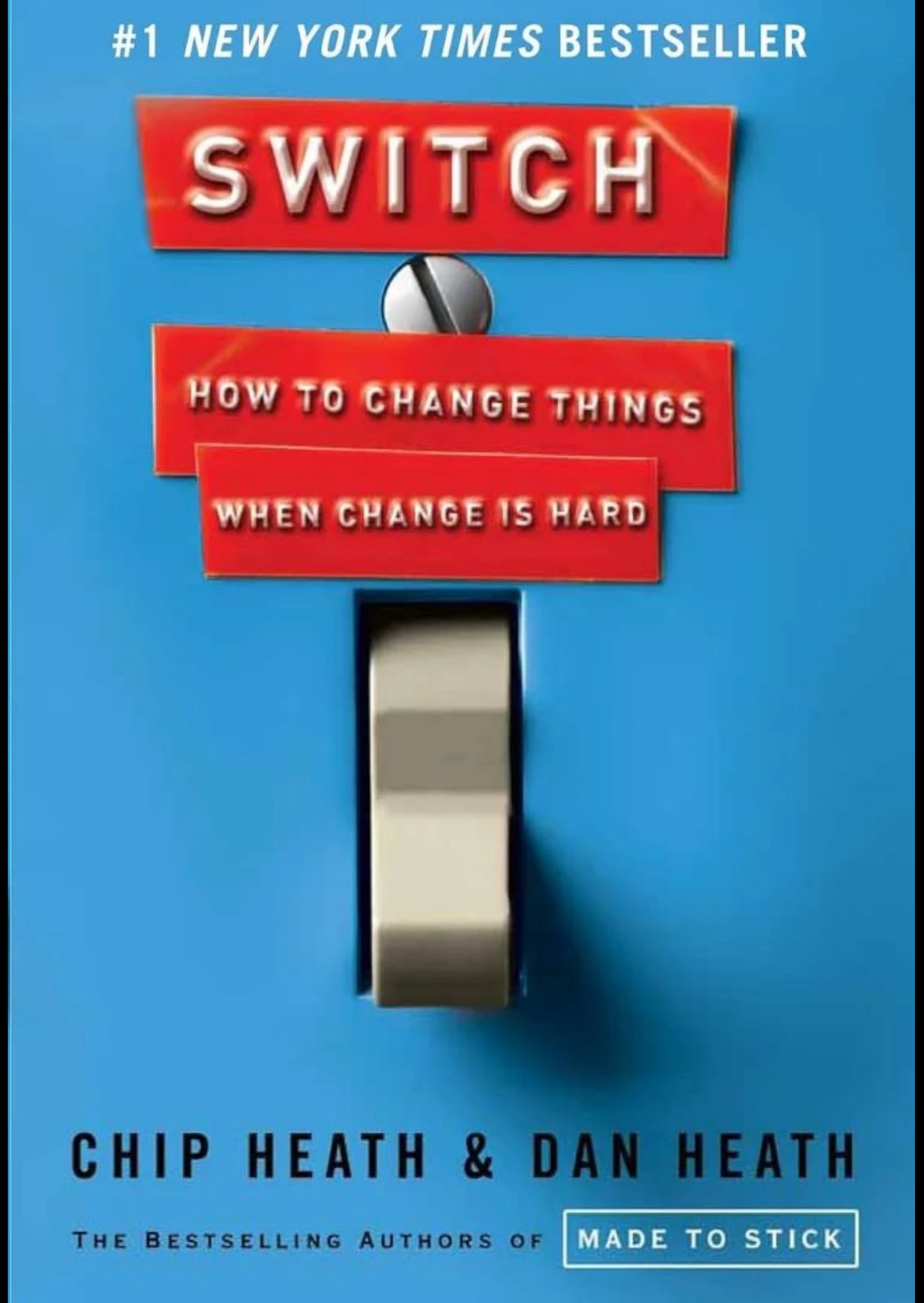
Fix It Now Or Delete It only works once you reach zero bugs.

But we had no idea how long fixing all bugs would take.

Why you can’t do this too

Your team probably hasn’t got all of these:

1. enough pain that your team attributes to bugs,
2. the agency to fix or delete the whole bug backlog,
3. a culture of quality that rejects buggy releases,
4. a product manager who understands that you can’t test product hypotheses with buggy releases
5. sales & marketing who don’t only want new features.



Switch:

how to change things

when change is hard

Chip Heath & Dan Heath

(2010)

Zero-bug policy summary

1. Getting started required support from the whole team.
2. Our motivation came from the pain of managing bugs.
3. Everyone had questions, and wanted to overthink it.
4. A shared vision for a better future helped (maybe).
5. The fix-all-bugs transition phase was hard work.
6. Once we got zero bugs for the first time, it all got easier.
7. We stopped worrying and talking about bugs so much 

↓ slides & articles

Peter Hilton

@hilton.org.uk

https://hilton.org.uk/presentations/zero-bug

Questions

How did we define bugs vs other kinds of change?

We didn’t.

We didn’t need a written definition,

because we evaluated one bug at a time.

What counts is whether we’re going to fix it.

You only need definitions for prioritisation games.

Nicholas Smith
@nicholassmith

I'm so bored of my project now I can't even bring myself to document and test it, there are no bugs, just surprise features

6:28 PM · May 21, 2009

1

What if we'd started with hundreds of open bugs?

Then we'd have to have considered the ☢ nuclear option ☢

DELETE YOUR BACKLOG

How did we prioritise for business stakeholders?

As product manager, prioritisation was *my* responsibility 😅

Other business stakeholders never wanted to discuss bug prioritisation.

If I did end up in a prioritisation discussion:

I'd talk about business outcomes, such as customer sales, customer satisfaction, average order value, etc.

Did we worry about slipping back into many bugs?

Yes, a little, but less for each month it still didn’t happen.

I sometimes reminded the team that we still fixed bugs first.

We avoided big releases that might cause many bug reports.

Feature flags decouple deployments from releases.

Did we refactor, and improve testing, while fixing?

Yes – much more unit test and integration test coverage.

Better code reviews.

Intermittent web shop checkout failures:

root cause identified,

redesigned to prevent inconsistent state on failure.

If we delete a bug report, might a series of small bugs compound and become a big problem?

Didn’t happen.

After all, we only deleted 9% of bug reports.

Deleting several related bug reports would be… unlikely.

Why were there so many bugs to fix?

Context: early-stage startup soon after public launch

QA wasn’t good yet

We hadn’t worked on reliability