Videos
Podcasts
Extra Crunch
—
Events
Advertise
Crunchbase
More

Cybersecurity 101
Fintech
Transportation
Google

Search

# JIRA is an antipattern

Jon Evans @rezendi / 3 months ago

Comment

**Atlassian's** JIRA began life as a bug-tracking tool. Today, though, it has become an agile planning suite, "to plan, track, and release great software." In many organizations it has become the primary map of software projects, the hub of all development, the infamous "source of truth."

It is a truism that the map is not the territory. Alas, this seems especially true of JIRA. Its genesis as a bug tracker, and its resulting use of "tickets" as its fundamental, defining unit, have made its maps especially difficult to follow. JIRA[1] is all too often used in a way which makes it, inadvertently, an industry-wide "antipattern," i.e. "a common response to a recurring problem that is usually ineffective and risks being highly counterproductive."

One thing that writing elegant software has in common with art: its crafters should remain cognizant of the overall macro vision of the project at the same time they are working on its smallest micro details. JIRA, alas, implicitly teaches everyone to ignore the larger vision while focusing on details. There is no whole. At best there is an "Epic" — but the whole point of an Epic is to be decomposed into smaller pieces to be worked on independently. JIRA encourages the disintegration of the macro vision.

What's more, feature-driven JIRA does not easily support the concept of project-wide infrastructure which does not map to individual features. A data model used across the project. A complex component used across multiple pages. A caching layer for a third-party interface. A background service providing real-time data used across multiple screens. Sure, you *can* wedge those into JIRA's ticket paradigm … but the spiderweb of dependencies which result don't help anyone.

Worst of all, though, is the endless implicit pressure for tickets to be *marked finished*, to be passed on to the next phase. Tickets, in the JIRA mindset, are taken on, focused on until complete, and then passed on, never to be seen again. They have a one-way life cycle: specification; design; development; testing; release. Doesn't that sound a little … um … waterfall-y? Isn't agile development supposed to be fundamentally *different* from waterfall development, rather than simply replacing one big waterfall with a thousand little ones?

Here's an analogy. Imagine a city-planning tool which makes it easy to design city maps which do include towers, residential districts, parks, malls and roads … but which doesn't easily support things like waterworks, sewers, subway tunnels, the electrical grid, etc., which can only be wedged in through awkward hacks, if at all.

Now imagine this tool is used as a blueprint for construction, with the implicit baked-in assumption that a) the neighborhood is the fundamental unit of city construction b) cities are built one neighborhood at a time, and neighborhoods one block at a time. What's more, one is incentivized to proceed to the next only when the last is absolutely complete, right down to the flowers growing in the median strips.

*Now* imagine that the city's developers, engineers and construction workers are asked to estimate and report progress purely in terms of how many neighborhoods and blocks have been fully completed, and how far along each one is. Does that strike you as a particularly effective model of urban planning? Do you think you would like to live in its result? Or, in practice, do you think that the best way to grow a city might be just a little more organic?

Let’s extend that metaphor. Suppose you began to build the city more organically, so that, at a certain significant point, you have a downtown full of a mix of temporary and permanent buildings; the skyscrapers’ foundations laid (i.e. technical uncertainty resolved); much of the core infrastructure built out; a few clusters of initial structures in the central neighborhoods, and shantytowns in the outskirts; a dirt airstrip where the airport will be; and traffic going back and forth among all these places. In other words, you have built a crude but functioning city-in-the-making, its skeleton constructed, ready to be fleshed out. Well done!

But if measured by how many blocks and neighborhoods are absolutely *finished*, according to the urban planners’ artistic renditions, what is your progress? By that measure, your progress is *zero*.

So that is not how JIRA incentivizes you to work. That would look like a huge column of in-progress tickets, and zero complete ones. That would look beyond terrible. Instead, JIRA incentivizes you to complete an entire block, and then the next; an entire neighborhood, and then the next; to kill off as many different tickets as possible, to mark them complete and pass them on, even if splicing them together after the fact is more difficult than building them to work together in the first place.

(If you prefer a smaller-scale model, just transpose: city → condo building, neighborhood → floor, block → unit, etc.)

And so people take tickets, implement them as written, pass them off to whoever is next in the workflow, consider their job well done, even if working on scattered groups of them in parallel might be much more effective … and without ever considering the larger goal. “Implement the Upload button” says the ticket; so that is all that is done. The ticket does not explain that the larger goal of the Upload button is to let users back up their work. Perhaps it would actually be technically easier to automatically upload every state change, such that the user gets automatic buttonless backups plus a complete undo/redo stack. But all the ticket says is: “Implement the Upload button.” So that is all that is done.

All too often, the only time anyone worries about the vision of the project as a whole is at the very beginning, when the overworked project manager(s) initially deal(s) with the thankless task of decomposing the entire project into a forest of tickets. But the whole *point* of agile development is to accept that the project will always be changing over time, and — albeit to a lesser extent — for multiple people, everyone on the team, to help contribute to that change. JIRA has become a tool which actually works against this.

(And don't even get me started on asking engineers to estimate a project that someone else has broken down, into subcomponents whose partitioning feels unnatural, by giving them about 30 seconds per feature during a planning meeting, and *then* basing the entire project plan on those hand-waved un-researched off-the-top-of-the-head half-blind guesses, without ever revisiting them or providing time for more thoughtful analysis. That antipattern is not JIRA's fault … exactly. But JIRA's structure contributes to it.)

I'm not saying JIRA has no place. It's very good when you're at the point where breaking things down into small pieces and finishing them sequentially does make sense. And, unsurprisingly given its history, it's exceedingly good at issue tracking.

Let me reiterate: To write elegant software, you must keep both the macro and the micro vision in your mind simultaneously while working. JIRA is good at managing micro pieces. But you need something else for the macro. (And no, a clickable prototype isn't enough; those are important, but they too require descriptive context.)

Allow me to propose something shocking and revolutionary: *prose*. Yes, that's right; words in a row; thoughtfully written paragraphs. I'm not talking about huge requirements documents. I'm talking about maybe a 10-page overview describing the vision for the entire project in detail, and a six-page architectural document explaining the software infrastructure — where the city's water, sewage, power, subways and airports are located, and how they work, to extend the metaphor. When Amazon can, famously, require six-page memos in order to call meetings, this really doesn't seem like too much to ask.

Simply ceasing to treat JIRA as the primary map and model of project completion undercuts a great deal of its implicit antipatternness. Use it for tracking iterative development and bug fixes, by all means. It's very good at that. But it is a tool deeply ill-suited to be the map of a project's overall vision or infrastructure, and it is *never* the source of truth — the source of truth is always the running code. In software, as in art, the micro work and the macro vision should always be informed by one another. Let JIRA map the micro work; but let good old-fashioned plain language describe the macro vision, and try to pay more attention to it.

---

[1] Atlassian seems to have decapitalized JIRA between versions 7.9 and 7.10, but descriptively, all-caps still seems more common.

**Image Credits:** Wikimedia Commons under a Public domain license.

View Comments (34)

**Sign up for Newsletters**

□ The Daily Crunch

□ The Weekly Roundup

□ Crunchbase Daily

See all newsletters

Email *

Subscribe

https://tcrn.ch/2QkOlik Copy

## Tags

Opinion

Atlassian

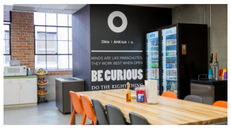
JIRA

TC

Atlassian JIRA

**Okta to acquire workflow automation startup Azuqua for $52.5M**

18 mins ago Ron Miller



**Samsung's very good Galaxy S10 is now on sale**

19 mins ago Brian Heater

### Changes at YC, $1.5B more for ride hailing, and Airbnb buys HotelTonight

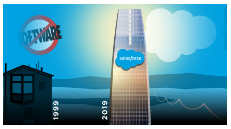
1 hour ago Alex Wilhelm, Kate Clark

### On #IWD2019, Safe & The City launches their new app for women's safety

1 hour ago Mike Butcher

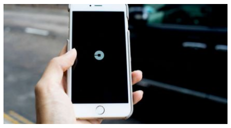
### Salesforce at 20 offers lessons for startup success

2 hours ago Ron Miller



### Uber pays $2.6M to settle historical charges it violated Dutch taxi laws

3 hours ago Natasha Lomas



### Cookie walls don't comply with GDPR, says Dutch DPA

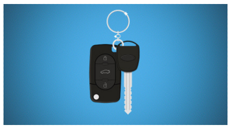
4 hours ago Natasha Lomas

### PolyAI scores $12M Series A to put its 'conversational AI agents' in contact centres

6 hours ago Steve O'Hear

### Car alarms with security flaws put 3 million vehicles at risk of hijack

8 hours ago Zack Whittaker



### Leica's Q2 is a beautiful camera that I want and will never have

14 hours ago Devin Coldewey

### Netflix star and tidying expert Marie Kondo is looking to raise $40M

15 hours ago Kate Clark

### Voatz, the blockchain-based voting app, gets another vote of confidence as Denver agrees to try it

15 hours ago Connie Loizos

### FCC 'looking into' reported error throwing broadband deployment numbers off by millions

16 hours ago Devin Coldewey

### YC's latest moonshot bet is a startup building a $380K 'flying motorcycle'

17 hours ago Lucas Matney

### Three imperatives for educators to get more women in tech

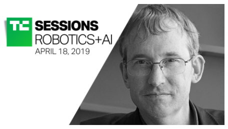
17 hours ago Melissa Kaufman

### Elton John's farewell tour marks the beginning for an audio augmenting wearable

17 hours ago Brian Heater

### Colin Angle will be speaking at TC Sessions: Robotics + AI April 18 at UC Berkeley

18 hours ago Brian Heater



### Instagram prototypes video co-watching

19 hours ago Josh Constine

### Facebook will downrank anti-vax content on News Feed and hide it on Instagram

19 hours ago Taylor Hatmaker

### WellSaid aims to make natural-sounding synthetic speech a credible alternative to real humans

19 hours ago Devin Coldewey

TechCrunch Privacy Policy About Our Ads Code of Conduct Terms of Service