Don't be a Vogon, make it easy to access your test data!

03 February 2025 · automation, exposure

The beginning of the hitch-hikers guide to the galaxy leads with an alien ship about to destroy the Earth, and the aliens saying we (mankind) should have been more prepared – as a notice had been on display quite clearly – on Alpha Centauri the nearby star system, for 50 years. Seriously, people - what are you moaning about – get with the program?

The book then continues with the theme of bureaucratic rigidity and shallow interpretations of limited data. E.g. The titular guide's description of the entire Earth is one word: "Harmless", but after extensive review the new edition will state: "Mostly harmless".

Arthur Dent argues with the Vogons about poor data access

This rings true for many software testing work, especially those with externally developed software, be that external to the team or external to the company. The same approaches that teams use to develop their locally developed usually don't work well. This leads to a large suite of shallow tests that are usually hard to maintain and often not trusted. The tests are usually an awkward and brittle addition like a cheap ill-fitting smart phone case, they half look the part but let's be honest – your phone might work better smashed than with that "indestructible" (and virtually unusable) case.

Let's take a quick step back and ask what's wrong? What's the problem that teams hit first? Usually – the first hurdle is getting that information. What information? Anything you need to

find out if there is a bug. Upon finding they don't have easy access to a data source (a DB, a log, an API etc) teams will usually start rationing access (let's not bother querying that…) or even avoiding getting whole swathes of relevant data. For example, they'll reduce the project to only accessing an app via the UI, or via limited APIs or Message Queues already provided.

This tendency to avoid the backend-data access is usually exacerbated by the monolithic nature of many externally bought software systems. (or the many in-house microservice based systems, that have ossified into a monolith)

Even if the team get that data access, they often assume that they are the only consumers and in fact they feel they should probably gate-keep that data. On the contrary its often better to make those data sources accessible via APIs or a convenient fast data store. Why? – so everyone so you can build more tests more easily.

It's not about the greater good, making the data accessible makes your life easier and has the added benefit that other teams get to test their integrations with your system more easily thanks to that handy API… The common anti pattern is to build for example a series of database queries into your test code directly. Much better to place that in a re-usable service that you can use in other tests, other teams can use, no matter their programming language (Yes, your team may use JavaScript or TypeScript to test your front-end but something else for the backend – and they all want to access the data!). Your test infrastructure is then loosely coupled and easily reusable.

These investments in data access and data generation are usually better than the diminishing returns gained from adding more slow and flaky tests via the user interface or hand rolling large quantities of XML or JSON test data – in each test.

Its not uncommon for teams to build large impenetrable test frameworks and huge test suites. The first step when you find yourself in this situation is not to build more or even rebuild again in the latest tools. It's to find out what data you really need to test and build around that guiding principle of open and easy access.