
On exploring crimes against software and causes for over-engineering.
Dependency hell

Click for more info

[Short 003] Crimes against software, Dependency hell
By Diogo Casado

2024/08/22 - Write & Record
2024/08/22-23 - Edit

Brainstorming

Leveraging existing libraries is necessary to produce software quickly. Trying to implement complicated standards will take time and effort that can distract us from working on the main functionality of our products. Working on things provided by dependencies (libraries or external services) can be way over our heads. And maintaining all the code necessary to keep up with changing standards in a time that most software is meant to be online is simply impractical.

I personally think that the less dependencies the better, and you might feel the same if you are an old school programmer. This feeling could come from a place of nostalgia from a time when most of the software was actually implemented by one person and all the heavy lifting was necessary. It was also a symbol of being a good programmer. Reassuring autonomy. It was craftsmanship. But most software was also offline and tangible. You would sell it and someone would own it. That thing was complete in itself. With the concept of SaaS most software moved online and is now an intangible thing from the user's perspective. And the more dependencies we as developers add to it, the less it feels like it's ours. It feels incomplete. And it could be made broken at any time. And maybe many of the dependencies we might add are just a consequence of complicating or

over-engineering the software and implementing features that are not really necessary. That's my opinion.

But this is not what this is about.

We will need to add dependencies to our projects. But when does a dependency become a bloat? It can be a deep hole. One package requires another. And another. And soon you have multiple points of failure that are simply out of your control. Sure the package could be open source and you could fix it. But that's the same as saying you can maintain software without any dependency. It's too much. And what about the security? There are many examples of bugs and even malicious contributions that jeopardized security. If it's a poison we have to drink, surely there must be a criteria to choose one that won't kill us. Meaning, a way to better choose dependencies. Pardon my lyricism, it's because I'm thinking of a crime narrative. Crimes we commit against our software.

Some thoughts:

- Dependencies that depend on too many other packages are sign of danger and we question the value that is being added by it;
- Choosing libraries that implement standards is safer than choosing libraries that just add "comfort";
- How often is the package maintained? How many people work on it?
- Is it an external service that can be run in a private structure? What is the cost of moving away from it? Are there alternatives?

Research

Dependency hell - Wikipedia

How one programmer broke the internet by deleting a tiny piece of code.

https://www.reddit.com/r/FlutterDev/comments/oplbly/how_many_are_too_many_packages/

Script

[Notes: I'm employing a crime motif. I'm thinking of the Unabomber guy with exploding packages for a funny reference. I find it funny to have his manifesto and the gnu manifesto being juxtaposed. It could also be some sort of terror movie with a bad Santa delivering gifts. Although less brainy.]

[🎥 Shot of me receiving a package from the side and opening it in front of the computer and getting surprised]

If you are creating software, you will need to choose packages and dependencies for your project. It will save time and effort. But it might also end up blowing up.. Your projects.

▶ From the archives: "Unabomber" Ted Kaczynski indicted on June 18, …

In his GNU Manifesto, Richard Stallman wrote:

[🎥 Capture from the source]
The fundamental act of friendship among programmers is the sharing of programs;

This is still a big influence on the open source community. In general, we want to use our skills and be of service to others.

But should we trust everything? When does a dependency become dangerous?

[🎥 Listening to the package (tic tac), foreshadowing?]

A software package can depend on another package that suddenly has a bug or a security issue. It might even disappear and break your project.

I usually pay attention to the signs that a package might blow up. What's the value I get from it? Is the guy maintaining the package called Ted? Or if it's an external service, is there an alternative just in case?

[🔙 I would return the package to my side]

Some packages are better left aside.. (idk I should be doing loop traps, it feels cheap)