**The Worst Kind of Programmer**



After 25 years of my career I came to understand that one particular type of programmers is the source of many problems in our industry. Here is the story of a project that was nearly destroyed by two such programmers. One of them was leading frontend development, the other – backend. While the rest of the team was slacking off because business requirements were in the works, these two chaps were working hard.

The frontend lead bootstrapped an Angular mono-repository with NX, Rx and other trendy technologies of the moment. The corporate environment did not lend itself easily to NX, but he pushed hard to solve or circumvent infrastructure problems by working with the domain expert. He set up a workflow that, while having Angular as the base framework, bore little resemblance to the official Angular tutorials.

The backend lead was no less motivated to bootstrap the Spring Boot project according to the corporate guidelines while adding a bit of a personal touch in the form of the Vavr library and a highly sophisticated hierarchy of JPA entities with multiple levels of inheritance, discriminators and generators. He then sprinkled it over with a hierarchy of validators based on booth Spring Bean Validation and a third-party validation framework and polished with a trendy testing framework.

The rest of the team observed them in admiration.

Months flew, the business started spewing requirements as best they could. The empty shells of both backend and frontend were already too complex for team members to work straight on business requirements, so both development leads worked hard to split business requirements into more manageable technical tickets while doing most of the grunt work along the way. Each of them worked harder than the rest of the team combined. When junior members were asking for tasks, they were given "less important work" such as API design or testing.

Then suddenly, the frontend lead quit for greener pastures. The business brought it an expert frontend developer who managed to keep on delivering for a while, then another one who lasted a bit longer. Junior devs were kept underused.

The load on the backend lead was ever increasing. He spent long days and even longer weekends delivering what seemed like simple CRUD interfaces. But they were so convoluted on the inside that little could be done to alleviate his workload. A few months later, the backend lead quit as well, and the team continued to work at snail's pace under the ever increasing pressure from the business to meet the deadlines.

The quality of code was failing to the point that Vavr objects were used in null comparisons and Angular Typescript was randomly intermixed with plain JS. At the end, the team barely delivered a product full of bugs. The future's looking bleak. None is up to the task to rewrite the many thousands of lines of code produced by the initial lead developers. Turnover is high. Costs are sky-high.

Does this description of a software development project feel familiar?

It does for me, as I have seen probably a dozen similar projects in different industries from Online Media to Public Services. The programmers I describe are usually considered to be the best of us, but I came to perceive them as the worst. Here is a non-exhaustive list of qualities of these people. Watch out for them:

**Ability to find satisfaction in solving abstract problems**

Great maths skills, glorious leetcode profiles, an aptitude to solve crosswords and compose puzzles are signs that a person can work on problems without real-life utility. They concentrate on the process, not on the outcome.

**Ability to put up many hours of work**

One has to be healthy and dispose of large swathes of time that they can dedicate to work. Family, kids, carpal syndrome are all signs that you are not of their kind.

**Passion in software engineering**

A bright and motivated programmer can always find a way to fit a technology he enjoys at the moment inside the project that pays his bills. Business is dull, coding is easy, so why not make it slightly more enjoyable by dragging in the latest tech everyone is excited about. Moreover, this will be yet another trendy item on the resume.

**Narcissism and self-confidence**

This is probably related to the Duning-Kruger effect in that the worst of our kind are usually relatively young, bright people in their late twenties or early thirties. They are consistently praised as overachievers and do not encounter much criticism.

**How not to handle the worst kind**

Appealing to management does not work. After all, the management in business environments is usually non-technical, and the worst kind are the best performers. Who should the management listen to if not to the best performers? Their performance is quantifiable, and team spirit is not.

Talking directly to the worst kind also does not work. They listen and kindly respond to the critique of their engineering choices, but the discussion of teamwork falls on deaf ears. They can not adapt their engineering choices to the least experienced team members, just like some adults can not adapt their speech when addressing kids.

**What can be done**

The first step to tackling the problem is to recognize it exists. The second is to spell it out: most software is written by teams, it has to be approachable by every team member. The third is to look around for existing ideas to tackle the problem. Surprisingly, many contemporary ideas in software engineering can be viewed as ways to fix the problem of the best of us:

**Golang, Lua and other simplistic programming languages**

There is broad consensus that Golang is simple to the point of being symplistic. It is in a way opposed to Rust as the means to an end and not a subject of much discussion. Golang teams strive, Rust teams… rust, because the language encourages concentration on itself, not on the outcomes of engineering projects.

**Scrum**

One largely unappreciated aspect of Scrum is the interchangeable nature of team members. While there may be specialisations, team members are expected to handle any task from the sprint. This should lead to simple engineering solutions, which are at the reach of all team members and not just the most sophisticated ones.

**DevOps**

Sophisticated programmers usually work in niche environments. It is relatively easy to be a sophisticated Spring Boot programmer or a sophisticated С++ programmer. DevOps turns everyone into generalists: configuring servers or cloud environments, monitoring, deploying, setting up pipelines while coding microservices. The sheer number of required skills transforms even the best of us into occasional noobs, fostering compassion towards less skilled team members.

**The End**

*The best of us are the worst.*