Does Your Code Hurt?

Help the team to write it better, then!

JetBRAINS Dino Esposito

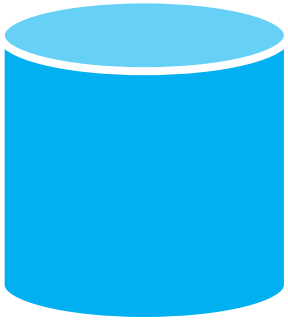
Why Do Software Projects ever fail?

It's always management.
It's almost never code.

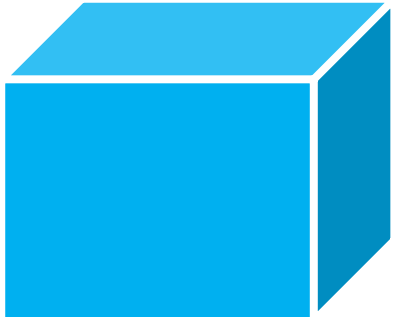
It works on my machine.

This is the sad true story of a crazy architect and a way too shy customer.

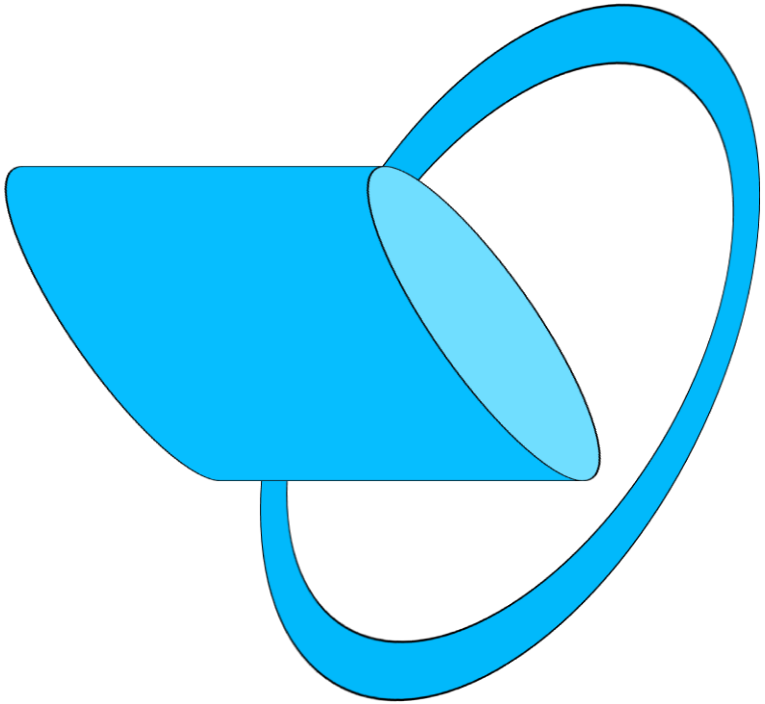
Customer:
I think our company wants and needs this.

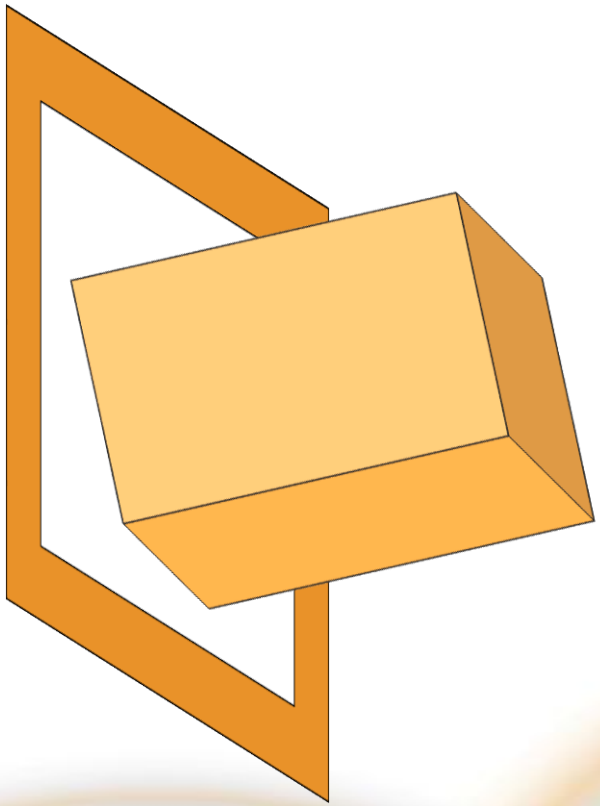


Architect:
It's far better if I build this for you.



The real-world domain:
Hold on! I'm like this.





Both systems worked according to **abstract** requirements

- The architect built a system that worked.
- But not the system actually requested.
- The customer had to change its business processes for the system to be useful.

The customer complained a lot of times

- The architect said “you’re the customer”.
- The architect said “speak abstract, I know what you need”.

The customer put pressure.

- The architect felt pressure and cut down development time.

The final system is painful to use.

1. Have Tools Assist Coding

- You **never** have time to **tidy up** your code.
- Tools **help** with auto-completion, live templates, idiomatic checks, IntelliSense, custom checks
- Unused references
- Dead code
- All is two or three clicks away

2. Tell People When Their Code Is Bad

- Don’t hurt but **don’t get hurt** as well
- Just ask **why** it was written the way it is
- Learn about motivation
  - Misinformation, bad attitude, limited skills or perhaps constraints you just didn’t know
- Attitude to challenge and be challenged.
- Be curious, gain evidence. Ruthlessly attack.

3. Make Everyone a Better Developer

- Address the code, not the coder.
- But try to improve the code through the coder.
- Any code can be fixed but if you don’t fix the developer you keep on getting bad code.
- Happier developer, richer team, more assets for the company.

4. Inspect Before You Check-In Code

Peer programming since 1998.

5. Projects Don't Need Heroes

04:00am – October 1999

5. Projects Don’t Need Heroes

- Developers love to work 80+ hours a week
- Should not happen. Never.
- Sometimes deadlines are unfair since the beginning. In some other cases, deadlines prove wrong along the way.
- Need of heroes stems from **unfair** deadlines.

6. Encourage Practice

6. Encourage Practice

- Roger Federer trains at least one hour a day. Developers don’t.
- Going regularly back to the foundation of object-orientation, design patterns, coding strategies, certain areas of the API helps keeping knowledge in the primary cache and more quickly accessible.

7. Ready to Refactor

- A software project begins following an idea or a rather vague business idea.
- Most software projects are just like moving targets and requirements are what move the target around
- To be effective, every developer should constantly be **ready to refactor**.
- Refactoring is one of those things that are hardly perceived as bringing value to the project.
- Bad luck that failing on refactoring takes value away.

In the context of a project with a significant lifespan and business impact, writing bad code ends up being more expensive than writing good code.

A project dies of bad code when the cost of working with code—that is, creating, testing and maintaining it—exceeds the cost that the business model can bear.

By the same token, no projects would fail for code-related issues if companies can manage to keep the cost of code extremely low.

Managers just **cut development** costs, hire cheap developers and ask/order to cut off tests and documentation.

They lower the **cost of producing code** and take the project home.

Unfortunately, producing *code-that-just-works* is only one aspect of the problem.

At the end of the day...

- Good coding is **more expensive** than producing *code-that-just-works*

- But it's **far cheaper** than maintaining and evolving *code-that-just-works*.

Thank you!