



10 Facts About Technical Debt

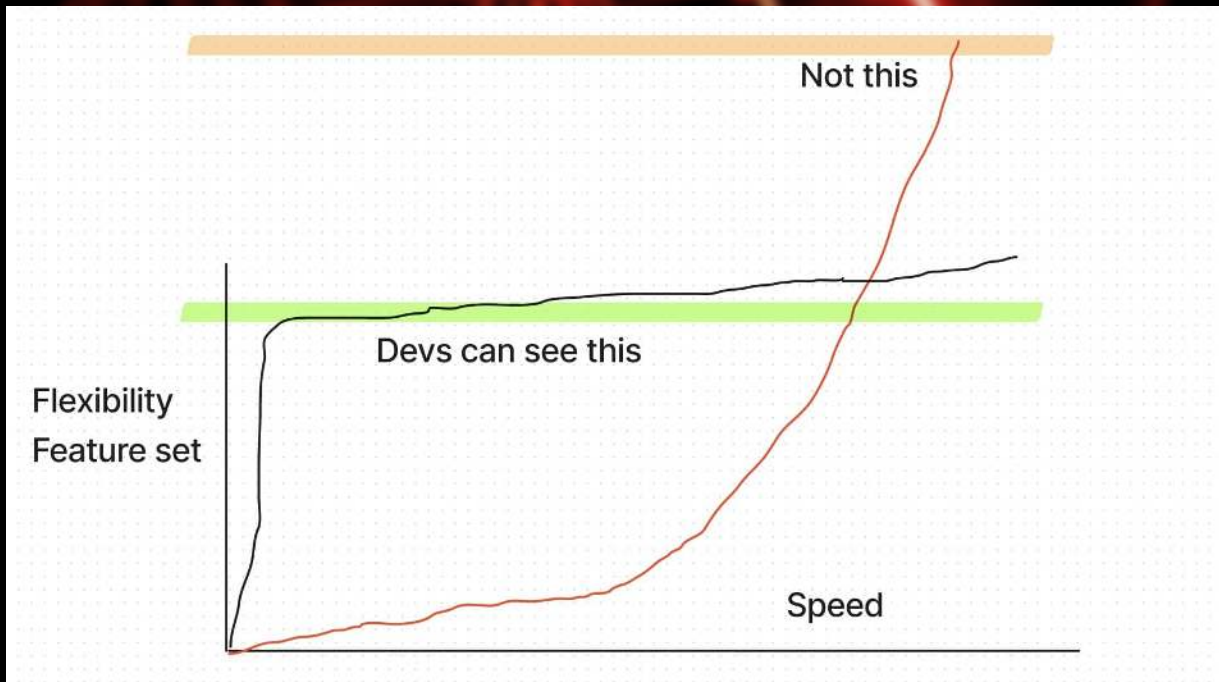
#1 Unreasonable: trying to prevent it

- It can even be beneficial in the early stages of a product
- Answer: what features are actually needed?
Then cut corners

Getting to market faster -> TD can be a good trade-off

#2 It happens unknowingly

Most of the time devs don't produce it on purpose. Reason: they cannot even see where the software will go in the future.



Why?

- Inexperienced developers may skip tests, which increases the time for any small change
- Experienced developers may build structures that are less flexible but faster, creating technical debt
- Technical debt happens with tests, specs, and good management because the desired flexibility is not known at the start

#3 It's always there and always needs to be dealt with

Stripe engineers spend 33% of their time dealing with technical debt

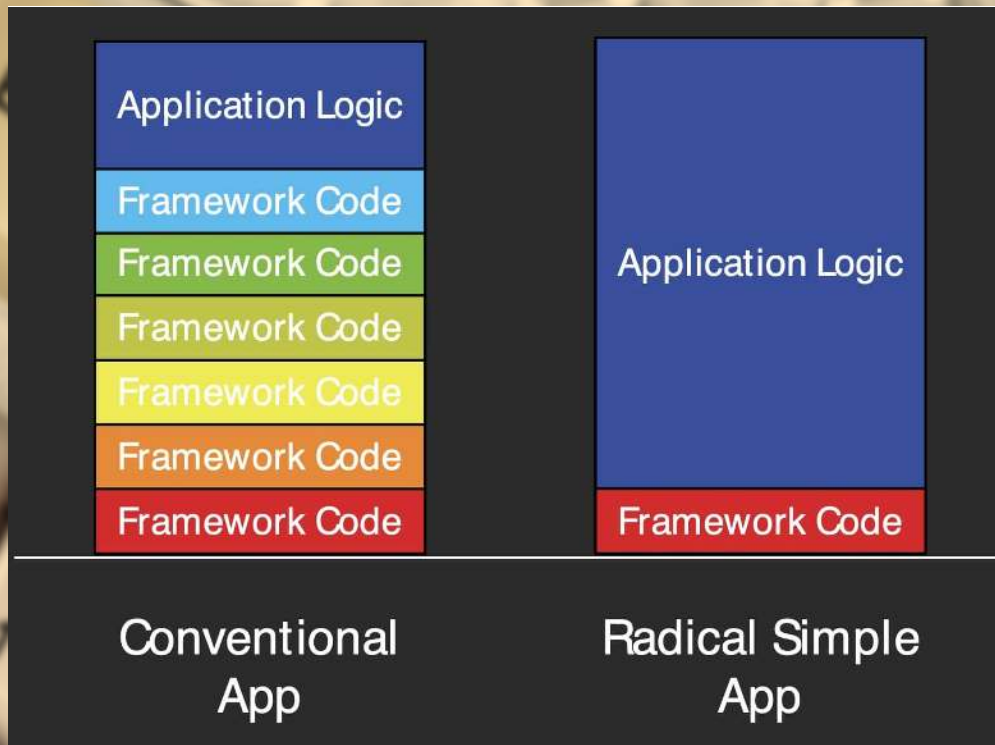
- We keep a separate swim lane in Jira for technical debt

other ideas to deal with TD include:

- developer Fridays, Ranger teams, DVD person dedicated for refactoring.
- For bigger thigs: Write TODOs in the code + dedicated tickets

#4 Write Less Code

- One of the best ways to minimize debt
- Rely on frameworks much

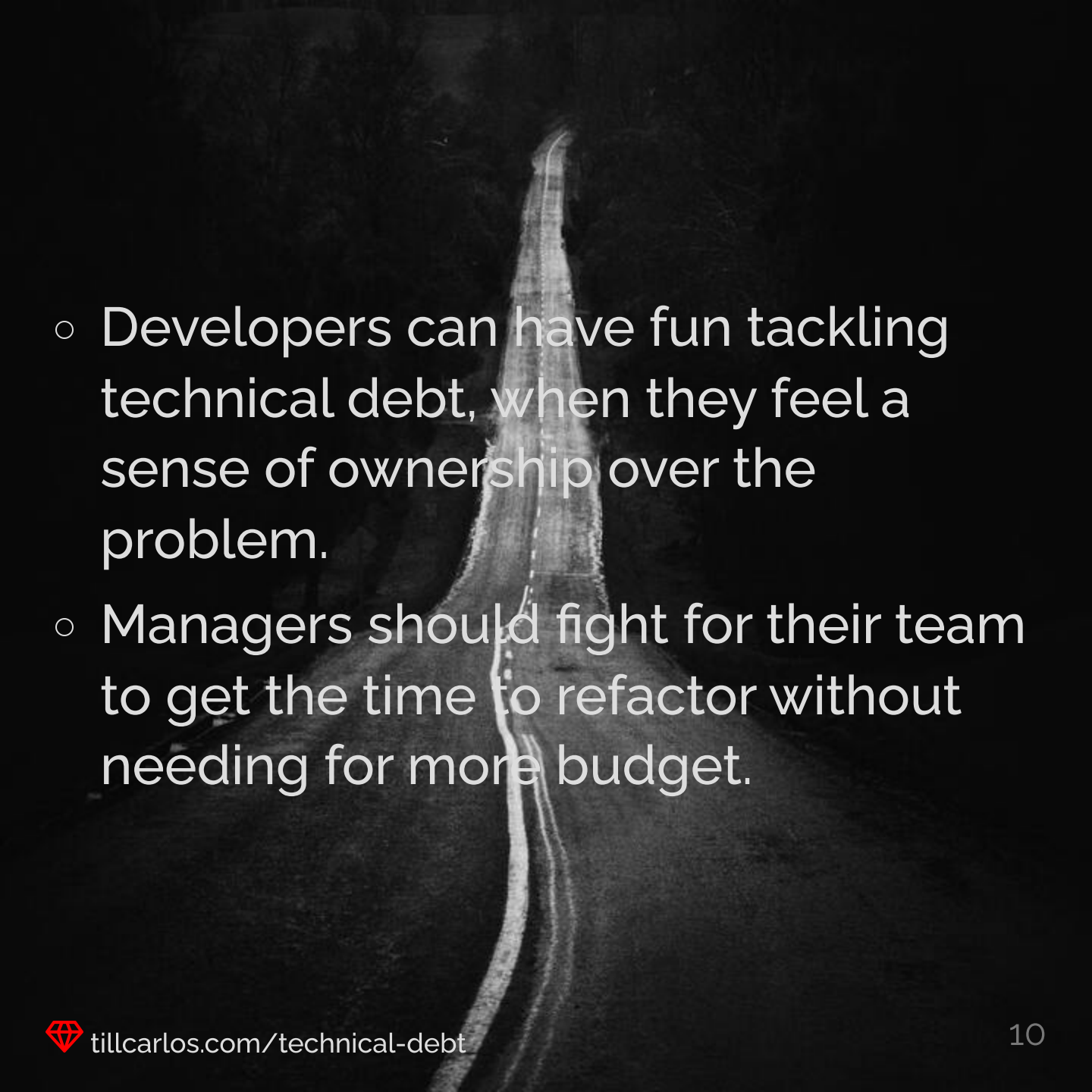


#5 Incentives → outcome

- Developers can be distracted by new tech
- Projects tend to expand to their deadlines and complexity increases with time, and ultimately results with technical debt.

#6 Team's Morale can diminish if you don't address it

- Technical debt doesn't usually scare good developers.
- Developers experience degrades with increased technical debt - leading to further issues

- 
- Developers can have fun tackling technical debt, when they feel a sense of ownership over the problem.
 - Managers should fight for their team to get the time to refactor without needing for more budget.

#7 Agile won't be the remedy

Agile helps with technical debt if there's room in each sprint to tackle it

- Waterfall model is still a good option for planning large software projects, where you know exactly what you want to build.

#8 Testing + Refactoring should be in the DNA of your team



Testing, Refactoring, Monitoring
belongs into the DNA of the team

Steve McConnell's in "Professional
Software Development":

We call this the mid-way between
full upfront design and planning
and having to constantly rewrite
entire software implementations
of systems. That's where the good
modular design of components
and containers comes into play.

#9 Clients need education

managing the technical debt requires strong leadership towards the client.

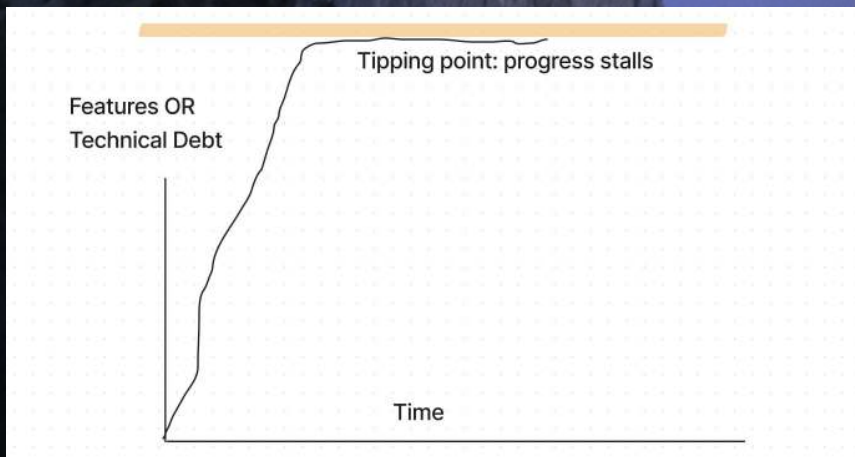
- often clients are not willing to “pay extra” and don’t see the value in reworking an already written subsystem.
- What needs to happen is:

- Plan refactoring into the budget
- Document every time you pile up technical debt
- Work through it constantly.

#10 Tipping point

There are projects where too much technical debt slows down development so much that no real progress is done, leading to:

- More managers and meetings
- Good developers leaving



... and only two ways:

- A full rewrite. Or heavy investment in refactoring.
- Good managers monitor ticket throughput and developer happiness to test how close the project is to the tipping point.
- In the startup world, this point may come after Product-Market-Fit.

Get in touch

Full post at
TillCarlos.com.

I make this to help
people like you be more
successful with their
software projects.

Questions? [write me!](#)

[Download slides \(pdf\)](#)

 tillcarlos.com/technical-debt

