
July 2016

# Managing Technical Debt

**Mircea Zetea**
**Technical Manager @Spyhce**
**@mircea_zetea**

Spyhce®

July 2016

# Managing Technical Debt

**Summary:**

- **Purpose of presentation**
- **Some history around it**
- **Where does technical debt occur? At what levels and which are actually the most dangerous?**
- **What are the consequences? Let's talk about interest**
- **What are the type of projects where it actually matters?**

# What is Technical Debt?

# Managing Technical Debt: Defining

First defined by **Ward Cunningham** in the early ‘90 as a metaphor,
A concept in programming to reflect the extra development work required when easy/short term solutions are applied to the expense of the best overall solution. Direct relation to finance loans.

# Managing Technical Debt: Defining

First defined by **Ward Cunningham** in the early ‘90 as a **metaphor**,
A concept in programming to reflect the **extra development work** required when **easy/short term solutions** are applied to the expense of the best overall solution. Direct relation to finance loans.

**Bob Martin:** Messy code is not technical debt. Technical debt is a conscious decision that obliges you to **write even cleaner code** as it is the only way to pay your debt.

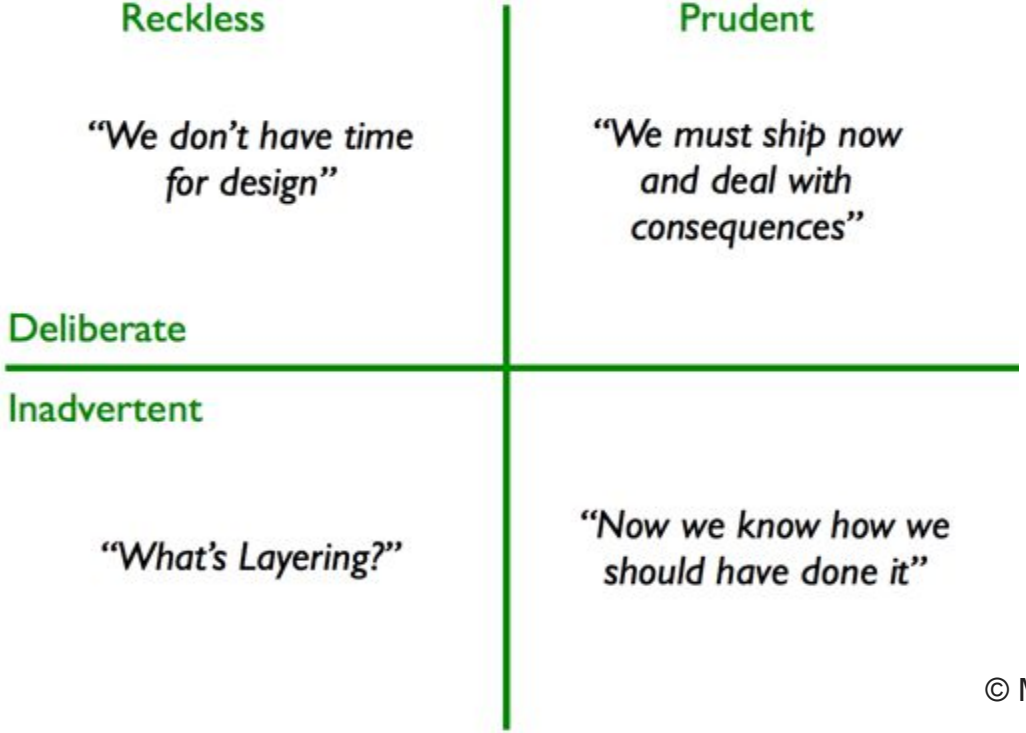
# Managing Technical Debt: Defining

First defined by **Ward Cunningham** in the early ‘90 as a metaphor,
A concept in programming to reflect the extra development work required when easy/short term solutions are applied to the expense of the best overall solution. Direct relation to finance loans.

**Bob Martin:** Messy code is not technical debt. Technical debt is a conscious decision that obliges you to write even cleaner code as it is the only way to pay your debt.

**Martin Fowler:** Both conscious and unconscious decisions are technical debt. If a mess is or not technical debt is the wrong question. The technical debt quadrant

# Managing Technical Debt: Defining




© Martin Fowler

# Managing Technical Debt: Defining

Why the quadrant?

Why approaching this way?

What can we learn from these 3 statements?

Why calling it “technical debt”? Why not … say… “Anti Patterns”

# Managing Technical Debt: Defining

Technical Debt is a “status quo”!

It is either there or it isn’t, no philosophy around it :) .

… And it’s a matter of acknowledging it exists (don’t live in denial!)

# Managing Technical Debt: Where

At what level does Technical Debt occur?

- Code level

# Managing Technical Debt: Where

At what level does Technical Debt occur?

- Code level
- Architecture Level

# Managing Technical Debt: Where

At what level does Technical Debt occur?

- Code level
- Architecture Level
  - Application Level
  - Infrastructure Level

# Managing Technical Debt: Where

At what level does Technical Debt occur?

- Code level
- Architecture Level
  - Application Level
  - Infrastructure Level
- Specification Level

# Managing Technical Debt: Interest

What exactly is the **“interest”**?

# Managing Technical Debt: Interest

How do we measure interest?

Is it different for the different levels (code, architecture, specs)?

# Managing Technical Debt: Manage it

Who is **responsible** for the Technical Debt and **when?**

# Managing Technical Debt: Manage it

Prevention: Code / Architecture

- Search / Documentation on previous things
- Ask for help
- POC it - if you have no idea how to build it. Remember, you will know how to build something once you've done it and it's not going to be the way you built it!
- Diligence - bare minimum for infrastructure but be ready to scale out (be aware of vertical vs. horizontal)

# Managing Technical Debt: Manage it

Specifications

- Communication, Communication, Communication - Specifications
- Know the difference between POC, POV and MVP. Make sure everyone knows it! Do not be fooled!
- Do not make assumptions. Famous last words: **What could possibly go wrong?**

# Managing Technical Debt: Manage it

What to do when you have it?

- Identify
- Know the problem(s) and its consequences. **Translate to cost!**
- Communication, Communication, Communication
- Plan to pay out the debt
  - See what you have to pay first and at what cost
  - Is it the **principal** or the **interest**?
- Collaborate. Seldom a single person is affected
- Test! Test! Test!

# Managing Technical Debt: Manage it

Know **who you are dealing with**: for companies / projects where delivering software is what they do TD is a lot more relevant than for companies where this is something on the side, just a helper.

Tech companies should put more value into this. If software is your business then you must care about technical debt!

Not so relevant for others. Sometimes manual tweaks are perfectly fine for a very very long time. Know the difference, Know the business, Know your boundaries!


4GIFs
.com