Technical Debt

What is it and how to manage it

Ermanno Scanagatta
github.com/ermannos
twitter.com/@escanagatta

What is technical debt

Extra time paid by developers struggling with problems in code

- readability
- maintainability
- extensibility
- testability
- usability

Shipping first-time code is like going into debt.
A little debt speeds development so long as it is paid back promptly with refactoring.
The danger occurs when the debt is not repaid. Every minute spent on code that is not quite right for the programming task of the moment counts as interest on that debt.

Ward Cunningam, 1992

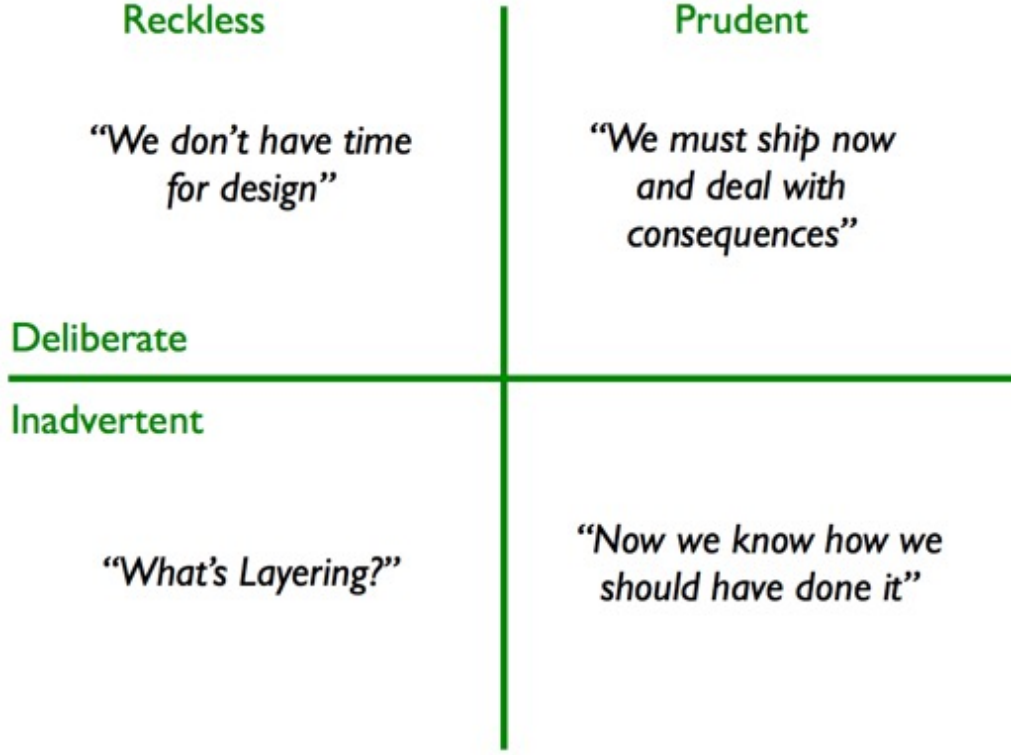
Is Technical Debt bad?

Not always.

TD permits to deliver business value **earlier** in time.

Sooner or later the team should **pay back** the accumulated debt.

It's important to **share the risk** with all stakeholders (team, management, business)



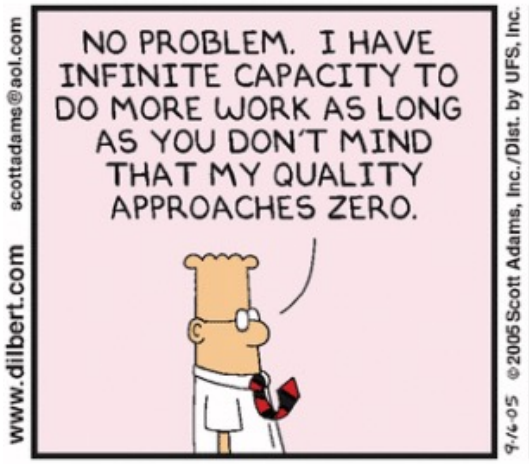
What is not technical debt

Not all software projects issues are Technical Debt

- Identified defects are not Technical Debt. They are **Quality Debt**
- Lack of process or poor process is not Technical Debt. It is **Process Debt**
- Wrong or delayed features are not Technical Debt. They are **Feature Debt**
- Inconsistent or poor user experience is not Technical Debt. It is User **Experience Debt**
- Lack of skills is not Technical Debt. It is **Skill Debt**

Measurements

- **Effort vs ideal time** implementing new features or doing maintenance
- **Time spent** to understand what software does
- **Time spent** to change contracts
- **Time dedicated** to remove it
- **Not objectively** measurable



How to analyze it?

- **Type of impact:** which activities and, abilities will it hurt?
- **Amount of impact:** how much will it hurt the business?
- **Duration and periodicity of impact:** will the impact happen once, or will it recur?
- **Timing of impact:** will the impact be perceived very soon or after delivery?
- **Age of debt:** is it “legacy TD,” or did the team add it during the last sprint?
- **Is intentional or not?** involuntary TD may trigger coaching and training for team members
- **Refactoring cost:** will help to plan and prioritize repayment activities
- **Dependencies with other debts:** is it included within the impact scope of another debt item?

How to reduce debt

NEW SOFTWARE

- Agile and iterative mindset
- TDD
- Indicators (code smells, SOLID)
- Retrospectives

LEGACY SOFTWARE

- Test coverage
- Refactoring

Some advice to deal with TD

- Avoid creating TD whenever possible
- Keep alignment with business
- Team should have a working agreement
- Plan the reduction of TD
- Education

The end

Thank you!

Questions?

References:

- Ward Cunningam - The WyCash Portfolio Management System
- Agile Alliance - Introduction to the Technical Debt
- Martin Fawler – Technical Debt