Technical Debt

Ángel Iglesias Préstamo – UO270534

Rosa García López – UO277921

Josué Fernández Argüelles – UO277912

As it seems, technical debt is a bill that companies must pay with time, money and resources for choosing speed over quality in their software. Other definitions like the *outsystems* one is: *Technical debt is the measure of the cost of reworking a solution caused by choosing an easy yet limited solution, robing your resources time, energy and the ability to innovate, adapt and grow*.

For instance, you can think of the shortcuts that we take to develop a feature on time. The feature is delivered in a sub-optimal way, so we need to go back in time to address those shortcuts and achieve a stable system or feature; that is the technical debt. In simple terms, it is the result of prioritizing speedy delivery over perfect code.

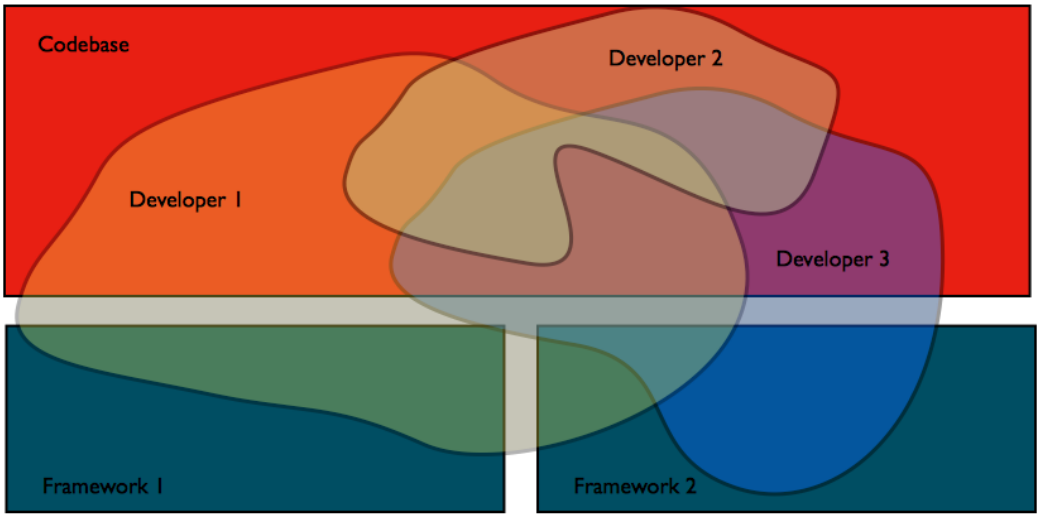
Types of technical debt

Technical debt can be broadly split into three types:

- **Unintended/accidental technical debt**: when designing a software system, a team tries to balance thinking ahead and future-proofing their design with simplicity and delivery. As the system evolves and requirements change, they might realize that their strategy is flawed or that the new functionality has become difficult and slow to implement.
- **Intended technical debt**: such as when a business and engineering manager agree that it is worth sacrificing quality and paying for refactors and code improvements later for the sake of launching more quickly.
- **Bit rot technical debt**: it happens overtime. A component or system slowly devolves into unnecessary complexity through many incremental changes, often when worked upon by several people who might not fully understand the original design.

Bus factor for measuring technical debt

We can use the bus factor for measuring the technical debt of our project. This metric determines which parts of the project are not shared among the team, but only for a part of it. So, in case those members leave it, the project will stall by the lack of knowledgeable or competent personnel. This may occur when the code is not readable. We will expand this topic later.



Causes

Some factors that may produce technical debt are:

- **Time Pressure**: Companies often release applications not full-featured due to pressure to deliver and accelerated timelines.
- **Constant change**: Full-featured applications completed on time may arrive in the market out of date.
- **Outdated Technology**: Modern applications require coding languages, frameworks and libraries that can become obsolete or not supported.

Symptoms of technical debt

The buildup of technical debt is a major cause for projects to miss deadlines, due to how difficult it is to estimate exactly how much work is necessary to pay it off. Some symptoms of technical debt are the so-called code smells. In other words, pieces of code that have an inferior quality than the rest of the application. Sometimes, it may be faster writing bad code – not following any design principle – than doing it properly. However, as the application grows modifying the features that were miscoded may be a nightmare. More on this in the following section.

Interest payments and snowball

When developing code that is not maintainable, the cost of writing some new features will increase as the technical debt does. That is called Interest payments. Not only that, but also the fact that this interest will increase along with the project, unless we refactor that piece of code that was not maintainable. This may cause a snowball effect: the higher the debt, the lower the motivation and love to the project, more bugs may appear, and the development process will take longer. In order us to fix that, we will have more time pressure, conflicts or needs for increasing the size of the team, increasing the debt once and again.

In the worst of the scenarios, refactoring will not be an option, making the project unmanageable. In fact, some companies have faced this kind of problems. One example may be Facebook, where developing a completely new virtual machine for it to be ran at their servers was more feasible than fixing the problem at source code. You can see more about this story at HHVM. This could have been solved from the beginning if they used another programming language in the server-side of their application.

Is technical debt bad?

Technical debt is not always bad. In fact, it is a part of every software development project, since tradeoffs between speed, cost, and quality are impossible to avoid. Technical debt mainly becomes an issue when ignored by business managers or accrued unintentionally through poor engineering decisions.

Sources

https://en.wikipedia.org/wiki/Technical_debt

https://medium.com/agileinsider/8-top-metrics-for-measuring-your-technical-debt-24e503b29529
https://www.outsystems.com/glossary/what-is-technical-debt/

https://www.getclockwise.com/blog/examples-of-technical-debt

http://www.leanway.no/competence-debt/

https://insights.sei.cmu.edu/blog/managing-the-consequences-of-technical-debt-5-stories-from-the-field/

https://www.projectpractical.com/technical-debt/

https://www.rootstrap.com/blog/what-is-technical-debt-with-examples/