What is Technical Debt?

During the whole software development process there are a lot of both technological and organizational/business decisions to be made. Proper decision making process should precede any actual work & execution to prevent existence of future problems and possible mistakes.

This works **only in theory.** In real life we rarely have the luxury to spend a proper amount of resources (including time) on this process. This preassure pushes us to agreeging to unavoidable tradeoffs and finally execution of many sub-optimal solutions (or sometimes not executing any solutions/changes when they are clearly needed).

Technical Debt is the constantly growing sum of “costs” of all those unavoidable, suboptimal decisions (or lack of decisions).

Intentionall / unintentional Causes of Technical Debt

Let’s say that we have to decide between development of a certain product using old, cheap and fast-to-code technology/solution “A” or newly invented but quite expensive and time-to-code demanding technology/solution “B”.

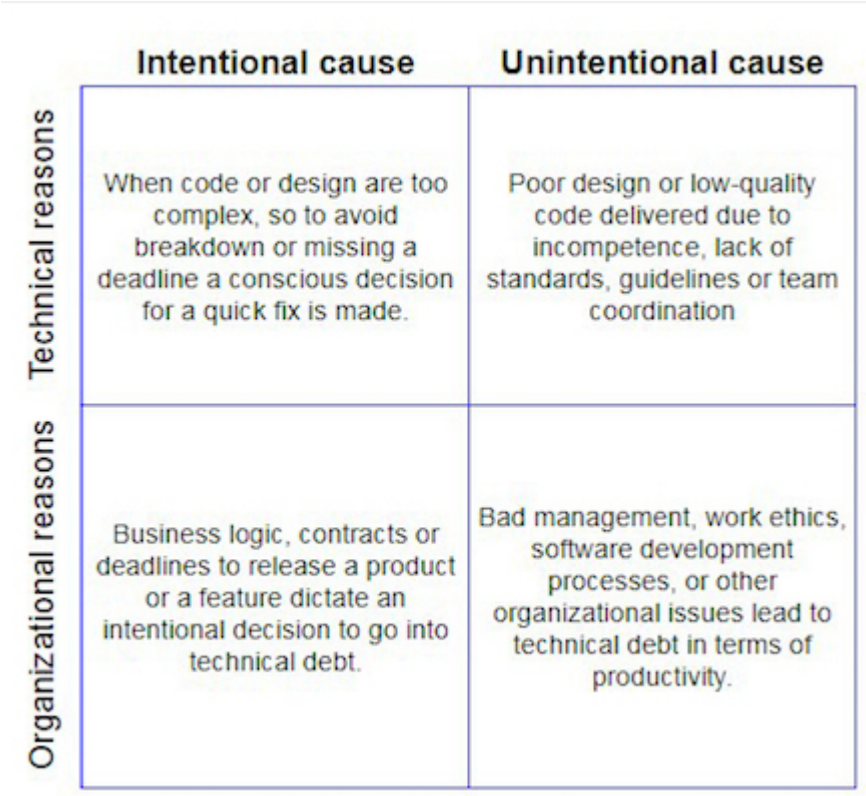
By Choosing A we will generate some savings in the short term but ultimately we will have to cover huge costs for either constant debugging (which will slow down new development) or completely rewrite old software using modern technology (which will potentially stop new development due to migration).

Choosing technology/solution B gives us long term stability and possibilities to grow but we pay for that with the higher price of software development.

Choosing either A or B means that we are **intentionally** accepting that our software will either accumulate Technical Debt faster or slower. In this case we are at least able to proactively address incoming problems.

On the other hand, lack of attention to decision making and poor quality of execution results in creation of **unintentional** technical debt that will occur regardless of choosing A or B. Hiring professionals and maintaining high

communication standards with them will help you to avoid it as much as possible.



	Intentional cause	Unintentional cause
Technical reasons	When code or design are too complex, so to avoid breakdown or missing a deadline a conscious decision for a quick fix is made.	Poor design or low-quality code delivered due to incompetence, lack of standards, guidelines or team coordination
Organizational reasons	Business logic, contracts or deadlines to release a product or a feature dictate an intentional decision to go into technical debt.	Bad management, work ethics, software development processes, or other organizational issues lead to technical debt in terms of productivity.

How to deal with a Technical debt?

1) Define it and keep track of it by visualization in Product Backlog.

“If something is not in the Product Backlog it does not exist”

This rule should apply not only to future development plans and small bugs but also to broader technical debt issues like for example:

“rewriting / refactoring code of feature X/Y/Z to comply with most recent product architecture changes in order to maintain highest application performance during rapid scaling of amount of end users“

2) Create a plan and execute it.

Once Technical Debt is visualised in Product Backlog you can focus on what is critical to “repay”. Take your time and discuss this from both a business (risk, value

to customer) and technical perspective (code performance issues). After defining where help is needed the most you have two solutions:

a) slice the elephant into sprints (evolutionary method)

In most cases there is always a possibility of planning a sprint with attention to **both** new development and technical debt handling tasks. Yes – this means slower development but this also means more stable, predictable, safe and scalable development!

b) prepare exclusive maintenance and/or migration sprint (revolutionary method)

In cases where there is no sense to work step by step (for example a huge incoming risk of critical product performance issues) you should consider switching all attention to Technical Debt and stop any “new” development.

3) Prevent Technical Debt from accumulating

The more you focus on the quality of the:
– decision making processes (risk assessment etc.);
– communication with Your product developers (discussing internal procedures, being open to honest feedback about certain solutions etc.) ;

the less Technical Debt will accumulate.

If you find this article helpful consider looking for more of them in our Sailing Byte’s Knowledge Base!