Nicolas Jacobs

Dr Krawitz

CS3500

2/15/24

Technical Debt Rough Draft

Technical debt stands for the computer science and programming concept which highlights the negative impacts made by taking shortcuts or opting for quicker solutions during software development. In the article called “A Balancing Act: What Software Practitioners Have to Say about Technical Debt,” by Erin Lim, Nitin Taksande, and Carolyn Seaman, it states, “The participants commonly acknowledged that technical debt is essentially a balance between software quality and business reality. One participant described it this way: Some shortcut has been taken or some less-than-desirable implementation has been done or we've descoped something for one reason or another, and so we've acquired some amount of the work that can or should be done at some point in the future. The reason why I believe it comes into being is simple business situations…. The business reality forces us to make choices at points in time to be able to get the broader outcomes of delivering a solution” (Page 1). Envision you must develop a complex program for your own computer science course, and you have limited time to do it. Being under time pressure and delivering, you will go for the quick coding techniques that include copying and pasting codes without fully understanding them, bypassing proper documentation, and delaying testing processes. However, even becoming "technically debt-free" is an error as, with time, it can only destroy the final software product's reliability.

A significant issue that technical debt creates in software development is the accumulation of interest. The interest that gets accumulated is caused by unresolved issues and shortcuts taken in software development. The interest is money that the company is losing because a certain aspect of their software is having issues. I remember Doc Krawitz telling us that time is money for companies that hire developers to make improvements to or create software that allows them to function. A lot of business transactions and personal data of a company is stored in the software. Like everyone in the business spectrum of the economy, companies want to be making significantly more money than what they spend on hiring developers or other business expenses. There are many instances where the success of a business is significantly correlated with the smoothness and effectiveness of their software. Given that most of today's society is heavily reliant on technology and online access, this creates a stream where companies can promote the business that they are trying to sell to interested consumers. As an experienced online consumer, I would argue that what attracts my attention to a particular service or good is the software's ability to provide me with a quick, reliable, and safe experience. If a software does not provide me with this type of experience, then I may feel frustrated or fearful emotions that will negatively impact my perspective on it and will cause me to never interact with the software again. Companies understand that these online consumer reactions can be caused by inefficient and faulty software. This is why developers are essential for companies to ensure to keep the relevance and functionality of their business.

The reason technical debt is such a significant issue is because there is a limited number of professionals that have the skills to fix such problems. One area of professionals that I mentioned earlier that have the skills to fix such a problem is developers. However, there are other professionals that can fix such a problem like project managers, product owners, quality assurance engineers, designers, and operation teams. I will be focusing more on the developer side in relation to technical debt as that is a profession that I am aiming to become and have experience with through the various Computer Science courses that I take at Curry College. Developers are one of the most impactful (both negative and positive) professionals in creating and resolving technical debt. A way that developers directly impact technical debt is through their codebase. Developers write code to improve or create software for the company they were hired by. Depending on the size of a company there can either be an exceedingly small number of developers or an exceptionally substantial number of developers working on software. Developers work in teams where they are each individually instructed to complete a specific task in a project. When working in teams, communication is especially important for the success of a project. Each task assigned to developers must be like puzzle pieces that connect to one another. This is because when each developer is done with their assigned tasks then all the tasks come together to form one unique project. Every developer has a unique way of creating their codebase and has a different level of experience. Some developers are creative and can easily create a codebase for a project while other developers are less creative and must use a pre-existing codebase for a project. There are also some developers that are more hard-working and diligent while other developers that are not hard-working and like to cut corners at every chance that they receive.

In developing software for companies, problems start to arise when developers do not communicate well and are not on a similar experience and work ethic level. The problems are usually caused because of the creation of a code base in the developmental phase of software that is not clean, not maintained well, and does not have comments explaining the various parts and their functionality. If the code is not functional, clean, or well-explained, it might reach a point where it becomes difficult to understand or modify. As a result, other developers may struggle to join the team and contribute to or extend the software, especially if there are issues with it. Also, incomprehensive testing by developers can lead to undiscovered bugs or vulnerabilities that may cause certain problems in software reliability and safety. In the article called “Do we need to pay technical debt in blockchain software systems?”, by Qu, Y., Bao, T., Chen, X., Li, L., Dou, X., Yuan, M., & Wang, H, it states “Developers of blockchain frameworks are expected to develop software frameworks that application developers can directly deploy. These frameworks should be fully tested and can be deployed directly into the application systems. However, due to the requirements of the project deploying time, or the budget of the open-source project, etc., there may be code issues that have not been fully resolved in the stable version released. The metaphor, technical debt is used to describe this situation, which is first proposed by Cunningham in 1993 (Cunningham, Citation1992)” (Page 2031). These problems create technical debt for companies in our economy. Earlier in the essay I had discussed how Doc Krawitz was telling us that time is money for companies that hire developers to make improvements to or create software that allows them to function. Most developers like to get paid by the amount of time they spend solving a task rather than get paid a set amount for completing the task by companies. Companies are willing to spend a good amount of money for developers in hopes that they will create a codebase for their software that will allow them to make even more money and have it run will little to no errors. The technical debt occurs when companies spend money that money, but the codebase produced by their developers does not make them more money and has a lot of errors that cause them to spend even more money to hire new developers. The company will spend more money trying to fix the code than make money from it through online consumers. There are also instances where companies must rehire their old developers because their newly hired developers cannot understand the codebase.

The solution for technical debt is creating clean code that is functional and well-explained. As a developer you want to create a codebase that can be easily understood and modified when needed. Clean code is a matter of more than just a technical detail. It is a matter of ordered and easy-to-read programming. It is the essential component in the software world and demonstrates deep programming talent. It represents the technique of coding, where machines can understand but also non-machine programmers can relate to, which boosts collaboration and understanding among developers. Having clean coding be a practice among modern developers can go a long way in removing technical debt from companies in today's economy. Having the skill set in creating clean code as a developer will make you a more prioritized candidate for wealthy companies. Through my experience of writing clean code in my Computer Science classes at Curry College, I have come to realize that it is significantly easier to detect and find mistakes as the codebase is organized into classes that each have their own specific functionality. Also, creating stubs with methods for a task in a project is another way that technical debt can be reduced. Many developers tend to write code immediately when completing a task for a project without organizing the components needed to complete the task successfully. What commonly causes technical debt in codebases is that developers start writing based on an idea in their head without planning out the steps. Later, they may want to change their codebase because they have a better idea or because it did not work as expected. This can lead to messy code that starts off following one idea and then suddenly switches to a mix of other ideas. Such inconsistencies may not confuse the developer currently working on the code, but they can confuse other developers. When companies hire developers on large contracts, they should make sure that they are equipped and knowledgeable of clean code skills to minimize the chance of wasting time a resource for a code base that can barely be maintained and is not understandable.

Technical debt is an intricate problem that is connected to various activities in business, technology, and people. The technical debt, created to meet the demands of time and limited resources, reflects the burden of taking short-term advantages over long-term sustainability in software projects. Technological debt is reminiscent of financial debt in accumulating interest and may lead to higher costs, lower productivity, and worsening user experience. Today business success depends on technical solutions and effective and reliable software frameworks. In this technically advanced environment, technical debt management is necessary for those companies which want to be on top of the market. It becomes apparent that the main contributors are the professionals, particularly developers, who deal with technical debt. Therefore, collaborating, communicating, and continuously improving are the basic elements for the diminishing of the problem. Through initiating a culture of clean coding, strategic planning, and comprehensive testing, companies can do well during the age of digitalization and interconnection. Furthermore, investment in the ongoing training and devising the best practices in software development are the key points to reduce technical debt and stimulate innovation. Considering the challenges of the digital era, it is imperative that we focus on the continual debt payoff so that we have a modern, flexible, and user-oriented technology system.

Works Cited

* Qu, Y., Bao, T., Chen, X., Li, L., Dou, X., Yuan, M., & Wang, H. (2022). Do we need to pay technical debt in blockchain software systems? *Connection Science*, *34*(1), 2026–2047. <https://doi.org/10.1080/09540091.2022.2067125>
* Aldaeej, A., & Alshayeb, M. (2023). Familiarity, Common Causes and Effects of Technical Debt: A Replicated Study in the Saudi Software Industry. *Arabian Journal for Science & Engineering (Springer Science & Business Media* *B.V. )*, 1–19. <https://doi.org/10.1007/s13369-023-08596-w>
* Lim, E., Taksande, N., & Seaman, C. (2012). A Balancing Act: What Software Practitioners Have to Say about Technical Debt. *IEEE Software*, *29*(6), 22–27. <https://doi.org/10.1109/MS.2012.130>
* Ramasubbu, N., & Kemerer, C. F. (2014). Managing Technical Debt in EnterpriseSoftware Packages. *IEEE Transactions on Software Engineering*, *40*(8), 758–772. <https://doi.org/10.1109/TSE.2014.2327027>