REPORT 7

Chaotic Scenarios of Technical Debt:

A Speculative Approach Aimed at Software Sustainability

[ANONYMIZED REPORT – NO IDENTIFYING INFORMATION]

Abstract

This academic article explores technical debt in information systems, using the speculative design approach to imagine the future implications of current technological decisions. It examines how the accumulation of technical debt can lead to system failures, inefficiency, and increased maintenance costs, especially in companies that rely heavily on software. The article also proposes a tool for the financial evaluation and management of technical debt (FAGFDT) to identify and quantify the financial impact of technical debt, helping companies make strategic mitigation decisions and promote sustainable development practices. Through an in-depth analysis, the article highlights the importance of managing technical debt proactively to ensure the longevity, efficiency, and competitiveness of organizations in the digital era.

1. Introduction

Technical debt is a growing problem that directly impacts the sustainability and efficiency of software systems in modern organizations. It refers to design and implementation choices that, although allowing rapid deliveries, accumulate problems that need to be resolved in the future, negatively affecting system maintenance and evolution. The chaotic scenario that may arise if technical debt is not managed effectively highlights the urgency of robust strategies to mitigate it and ensure the longevity and competitiveness of companies in the digital market.

The importance of managing technical debt proactively is emphasized by studies that associate the accumulation of this debt with systemic failures, operational inefficiencies, and high maintenance costs. Through the analysis of speculative scenarios, this article highlights the potential negative consequences of unmanaged technical debt, illustrating how it can lead to system failures, increased costs, and loss of competitiveness. The exponential growth of technical debt can lead to a critical point at which companies find themselves unable to maintain and update their software systems, compromising their operations and their capacity to innovate.

This article uses the speculative design approach to imagine the future implications of current technological decisions, exploring how different technical debt scenarios can impact the sustainability of software development practices.

In this context, the adoption of new, more efficient procedures and the use of artificial intelligence emerge as promising solutions for managing technical debt. The application of these technologies enables the identification, quantification, and mitigation of technical debt in a more precise and effective manner, promoting sustainable development practices. Understanding the importance of managing technical debt proactively and the possibilities offered by artificial intelligence prepares us to face future challenges and ensure the continuity and success of software systems in organizations.

2. Theoretical Foundation

This section aims to conceptualize speculative design, technical debt, and how speculative design can be used to explore how current technological decisions may impact the future.

2.1. Speculative Design

Speculative design can stimulate debate and reveal choices that exist beyond the limitations of current business models and technological approaches. This term was popularized by Anthony Dunne and Fiona Raby [Dunne and Raby 2013]. They argue that design should not only solve problems but also question and challenge existing assumptions. They also suggest that speculative design allows us to imagine how the future can be radically different.

This imagination is conducted through the use of artifacts or tools designed to assist in projecting futures. Instead of focusing on solving immediate problems, this approach creates objects and narratives that symbolize future challenges. These speculative artifacts are used to provoke critical discussions about the direction toward these futures or about the creation of new realities. Speculative design is an effective approach to materialize and experiment with alternative visions of the future, facilitating the understanding of the possible consequences of our current technological choices [DiSalvo and Lukens 2013].

Within this approach, several methods are employed to explore and question these potential futures. Some discussed by Anthony Dunne and Fiona Raby in *Speculative Everything* [Dunne and Raby 2013] are highlighted:

- **Design Fiction:** Combination of fiction elements with design to create speculative future scenarios, helping to explore the consequences of technological innovations.
- **Critical Prototyping:** Development of prototypes that are not necessarily functional but serve to explore and question alternative futures and their implications.
- **Speculative Scenarios:** Creation of future scenarios based on trends and emerging technologies to explore possible impacts and ethical issues.
- **Worldbuilding:** Construction of detailed fictional worlds, complete with rules, cultures, and technologies, to explore the consequences of different future scenarios.
- **Future Narratives:** Creation of detailed stories and narratives that explore possible futures, helping to visualize how technological and social changes may unfold.

Each of these methods offers a distinct approach to imagining and exploring alternative futures, encouraging debate and analysis of the potential consequences of technological innovations.

2.2. Technical Debt

Technical debt refers to the consequences of design or implementation choices that are far from ideal, made with the intention of saving time. These choices, although they may allow faster delivery in the short term, accumulate problems and complexities that must be resolved in the future, negatively impacting system maintenance and evolution. Kruchten, Nord, and Ozkaya [Kruchten et al. 2012] define technical debt as a set of compromises that, when not properly managed, can lead to a gradual deterioration of software quality, resulting in performance issues, recurring bugs, and difficulty in implementing new features.

A very common approach to explaining the concept of technical debt is to make an analogy with financial debt [Cunningham 1992]. In the context of short-term versus long-term decisions, just as a person may take out a loan to obtain immediate benefits, such as buying something now and paying later, a company may choose a quick and non-ideal solution to create a system rapidly. However, in the long term, just as loan interest accumulates and must be paid, the “interest” on technical debt manifests as increased cost and effort in system maintenance and evolution.

The accumulation of interest in technical debt can be compared to the accumulation of interest in financial debt. In the case of financial debt, interest accumulates over time, increasing the total amount to be paid. Similarly, technical debt accumulates as unresolved problems complicate the system, making it more difficult and costly to modify and maintain. This includes an increase in bugs, degraded performance, and growing complexity.

To manage these debts, both financial and technical, payments are required. In the case of financial debt, the person must eventually pay the principal amount plus interest. To deal with technical debt, the company needs to “pay” through code refactoring, design improvement, resolution of accumulated problems, and so on. This process involves dedicating time and resources to correcting previous compromises that led to technical debt, ensuring the sustainability and quality of the system in the long term.

However, unlike financial debt, technical debt can also occur indirectly or unintentionally [McConnell 2013]. Indirect technical debt is a problem that was not identified at its origin. In other words, it was not planned and is usually identified a posteriori. It often arises unintentionally due to factors such as lack of knowledge, changes in the project context, or non-ideal decisions that are not immediately evident.

2.3. Technical Debt from the Perspective of Speculative Design

The association between speculative design and technical debt allows a deep exploration of the implications of current technological decisions for the future. Speculative design offers an approach to mapping the current state, using a detailed mapping of signals and trends present in technical debt management. When not properly executed, this can result in high maintenance costs and difficulties in software evolution [Li et al. 2015]. Through speculative design, it is possible to visualize these impacts and better understand how technological decisions made today can influence system development in the long term.

Speculating about possible futures is a crucial exercise provided by speculative design, allowing the anticipation of scenarios in which technical debt reaches critical levels. This process involves the creation of narratives that illustrate how technical debt can affect the sustainability of software development practices. For example, one can imagine a situation in which a company faces major operational and financial challenges due to the accumulation of technical debt, leading to discussions about the need for stricter management and mitigation policies. These speculative scenarios encourage critical debates about best practices and sustainability in software development, preparing teams to face future challenges in a more informed and strategic way.

The projection of a desirable future thus becomes an essential step to align current practices with long-term objectives in software development. Speculative design allows organizations to identify where they want to go, visualizing a future in which technical debt is managed efficiently and sustainably. This desirable future includes the implementation of robust review and refactoring processes, the adoption of agile practices that minimize the introduction of new technical debt, and the creation of an organizational culture that values code quality. By integrating speculative design into technical debt management, organizations can not only mitigate risks but also promote healthier and more sustainable software development.

3. Methodology

The study was developed through the analysis of future scenarios based on trends and emerging technologies to explore possible impacts related to the generation of technical debt. This was carried out based on the speculative design practice manual [Loutfi 2024], which proposes the use of three sequential stages distributed in the following notebooks: "Where are we?", "Where are we going?", and "Where do we want to go?".

3.1. Notebook 1 – Where Are We?

The investigation of technical debt using speculative design begins with the essential question: "Where are we?". This analysis involves a detailed description of the current context.

- **Definition of the Theme**: The first step involves the clear choice of the theme to be investigated, which in this case is the intersection between technical debt and speculative design. The precise definition of the theme is crucial to establish a coherent and grounded direction for the study.
- **Mapping the Current State**: Next, a detailed mapping of the current state of the theme was carried out. This involved the analysis of reports on the estimated cost of technical debt and scientific articles. A survey was also conducted to understand how people connect with technical debt. This allowed a visual representation of how the most relevant actors influence the generation of technical debt.
- **Mapping Signals and Trends**: Given the context of this study, an analysis of signals and trends was carried out for the scenario of how technical debt impacts companies, increasing costs and causing inefficiencies. The Google Trends tool was used to verify whether there is concern among companies regarding these factors.

3.2. Notebook 2 – Where Are We Going?

To answer the main question of this notebook, it is necessary to define a future date that will be used as the basis for speculative scenarios. This future scenario helps elaborate a set of cause-and-effect relationships that result in positive, negative, and neutral implications.

- **Defining the Future**: This notebook begins with the definition of the future timeframe for the speculative scenario. This work considers a period of 10 years for analyzing the future of technical debt.
- **Mapping the Future Scenario**: This involves describing how the trends identified in Notebook 1 would reconfigure the current scenario if no intervention were carried out. A reflection on cause-and-effect relationships is conducted. Due to the definition of a possible 10-year future scenario, only technologies from the Innovation Map with a Technology Readiness Level [Mankins 1995] above level 6 were considered:
    - Prototype Test: The prototype is fully functional and ready for testing in a relevant industrial environment.
    - Prototype Demonstration: The prototype is fully demonstrated in an operational environment.
    - Ready for Implementation: The technology is developed and qualified. It is readily available for implementation, but the market is not yet fully familiar with the technology.
    - Fully Operational: The technology is operational and demonstrates considerable competition in the market among manufacturing industries.

3.3. Notebook 3 – Where Do We Want to Go?

This notebook encompasses the projection of the IT solution to be developed in the future, its description, and its nature. In addition, it includes other actions that should be carried out to achieve this future.

- **Designing an IT Solution:** Imagining an IT solution that could be developed in the future to mitigate the negative implications identified in Notebook 2. Within the context of the study, the focus was on developing a tool for the financial evaluation and management of technical debt.
- **Other Actions:** In addition to the IT solution, an analysis was conducted of which technologies or procedures do not yet exist, whether legal changes would be necessary, and what other changes would be required or could support the desired solution. The greatest current difficulty in implementing the imagined solution is the difficulty in financially tangibilizing technical debt.

4. Results and Discussion

4.1. Mapping the Current Scenario in Which Technical Debt Is Embedded

Technical debt is influenced by several actors and their interactions. Each actor has a specific role, and their actions and decisions can contribute to the accumulation or reduction of technical debt. By understanding these relationships within organizations, it is possible to imagine potential scenarios of technical debt accumulation and management.

- **Clients:** Clients play an influential role by providing requirements and expectations that shape the scope and complexity of software projects. They often introduce scope changes during development, requiring rapid adjustments that may result in increased technical debt. Continuous client feedback is essential for identifying problems and adjusting development as needed. However, expectations regarding delivery speed and final product quality can pressure development teams to compromise code quality in favor of rapid deliveries, thus contributing to the accumulation of technical debt.
- **Executives:** Executives are responsible for making strategic decisions about business priorities, resource allocation,

and budgets, directly influencing technical debt management. They seek to maximize return on investment (ROI), which can lead to compromises that result in technical debt if not properly managed. Executives decide the available budget for maintenance and refactoring, critical areas for technical debt mitigation. Their strategic decisions and resource allocation largely determine the organization’s capacity to handle technical debt effectively and sustainably, balancing the need for innovation with maintaining software quality.

- **Project Managers**: Project managers are responsible for coordinating work, maintaining alignment between what will be delivered by technical teams and other stakeholders, such as clients and executives. They manage deadlines and resources, deciding which tasks are prioritized, including those related to technical debt repayment. Pressure to meet deadlines can lead to the introduction of temporary solutions that increase technical debt. Project managers also plan development cycles, balancing the need for new features with technical debt repayment. Their ability to prioritize tasks effectively and plan for the long term is vital to minimizing the negative impact of technical debt on the project.
- **Technical Team**: The technical team plays a crucial role in the creation and maintenance of code or solution design, being directly responsible for implementing new features and for the quality of the produced code. They frequently face tight deadlines that lead them to make trade-offs between quick and sustainable solutions, resulting in the accumulation of technical debt. In addition, they are responsible for paying technical debt, an essential practice to improve software quality and maintainability. Non-ideal decisions made by this team significantly influence the amount of accumulated technical debt.

Figure 1 was used to sketch a representation of the current technical debt scenario and to show the relationship between the involved actors and the role of each one.

Figure 1. Actors that influence technical debt

It is important to highlight that nowadays many companies are essentially software-based. They use advanced technologies to improve operational efficiency, deliver products and services to their clients, and create new business opportunities. Digital transformation has allowed companies from all sectors to integrate software into their daily operations, making it a crucial component for competitiveness and innovation. Modern software-based companies have the potential to reach high levels of efficiency, innovation, and growth. However, they must carefully address challenges related to technical debt to fully take advantage of the opportunities offered by digital transformation.

Figure 2 presents interest over time based on Google Trends.

Figure 2. Interest over time on Google Trends

The 2022 report from the Consortium for Information and Software Quality (CISQ) on the Cost of Poor Software Quality in the U.S. [CISQ 2022] highlights a significant increase in the estimated total cost of technical debt. The principal value of technical debt in the U.S. was estimated at USD 1.52 trillion. This number is approximately equal to the total amount spent on the entire U.S. IT workforce in 2022.

Although the principal value of technical debt is estimated, there is no good estimate of the accumulated interest value. Interest refers to the extra effort that technical teams must expend when making changes due to the existence of technical debt. This effort accumulates over time as software becomes more fragile. Technical debt not accompanied by effective management is a significant problem that tends to worsen if nothing is done.

Technical debt is not the only cause of inefficiencies and increased costs for companies, but its effective management is fundamental to solving part of these problems, especially in software-based companies. Some research was carried out using the Google Trends tool to analyze company interest in both topics over the last 20 years.

The obtained data indicate that the year 2024 presented the highest interest in the topic Business Efficiency, reaching a value of 100, which represents the peak popularity of the term.

Information was also sought on interest in the topic Cost Reduction, which also reached its peak popularity in that year. This significant increase in searches suggests that companies are increasingly seeking to be aware of the operational challenges they face.

4.2. Exploring Possible Future Scenarios for Technical Debt

Speculative design can be used to imagine future scenarios in which accumulated technical debt has significant consequences. Negative scenarios can be visualized in which the neglect of technical debt results in ineffective software, security vulnerabilities, and high maintenance costs.

- **Generalized Bankruptcies:** In a future where technical debt is ignored, many companies, especially software-based ones, face bankruptcy due to their inability to maintain and update their systems. The increasing complexity of code leads to unsustainable maintenance costs, leaving companies unable to compete in the market.
- **Ineffective and Vulnerable Systems:** Systems overloaded with technical debt become ineffective and riddled with failures. The lack of maintenance and updates makes these systems vulnerable to cyberattacks, increasing the risk of data loss and compromising information security.
- **High Maintenance Costs:** System maintenance costs increase exponentially, draining resources that could be invested in innovation. Companies spend more time and money fixing emergency problems than developing new features or products. Few companies manage to maintain extremely expensive IT structures.

Reflecting on this future, it is possible to identify cause-and-effect relationships (x → y) that exacerbate maintenance and innovation problems in this scenario:

- **Growth of Code Complexity (x) → Unsustainable Maintenance Costs (y):** Continuous neglect of technical debt results in a constant increase in code complexity, making it more difficult to understand and modify. The growing complexity leads to unsustainable maintenance costs, as each change requires more time and effort and introduces new errors.

- **Increasing Pressure to Deliver Features Quickly (x) → Compromised Code Quality (y):** Pressure to deliver new features quickly leads to shortcuts in development, compromising code quality. This increases technical debt and reduces long-term software sustainability.
- **Accumulation of Legacy Code (x) → Difficulty in Implementing New Features (y):** The accumulation of legacy code caused by the increasingly rapid evolution of technology complicates the addition of new features needed to meet market demands.
- **Misalignment between Technical Teams and Business (x) → Ineffective Technological Solutions (y):** Lack of communication and alignment between technical teams and business objectives results in technological solutions that do not meet real company needs, wasting increasingly scarce resources and efforts.
- **Underestimation of Technical Debt (x) → Delays and Budget Overruns (y):** Underestimating technical debt during project planning or execution leads to unrealistic schedules and budget overruns, as more time and resources are needed to deal with unexpected problems.

The implications of accumulated technical debt can vary. Negatively, company bankruptcy is one of the most severe consequences, especially affecting software-based companies that cannot deal with technical debt, resulting in unemployment and economic instability. Positively, the crisis may lead to the need to develop new technological solutions that address technical debt more effectively. Companies and developers may be motivated to create innovative tools and methodologies to manage technical debt proactively. In a neutral sense, stagnation in software development occurs when companies do not fail but also cannot innovate, remaining merely operational without significant growth.

Ignoring technical debt not only compromises operational efficiency and system security but can also lead to generalized bankruptcies, especially in software-based companies. Therefore, it is crucial for companies to invest in sustainable and innovative development practices to mitigate the negative impacts of technical debt and ensure their longevity and success.

4.3. Idealizing a Solution to Promote Sustainability

The alarming scenario that arises if no action is taken in favor of software sustainability—and, consequently, of the companies that base their operations, products, or services on it—highlights the urgent need for effective strategies to combat technical debt. The adoption of new, more efficient procedures, driven by the boom in artificial intelligence, offers a promising hope to mitigate these risks. Therefore, it is essential to manage technical debt proactively, preventing future problems and ensuring the longevity and efficiency of software systems. With this understanding, the proposed solution to face this challenge is now presented.

4.3.1. Proposed IT Solution: Financial Assessment and Management Tool for Technical Debt (FAGFDT)

The proposed solution is a financial assessment and management tool for technical debt (FAGFDT). This tool focuses on early identification of technical debt and on measuring its impact in financial terms. It provides quantitative insights to guide strategic decisions for technical debt management. This solution would work as follows:

- **Early Identification:**
    - The tool integrates with code repositories and version control systems to continuously monitor source code.
    - It uses predefined rules and code quality metrics, such as cyclomatic complexity, test coverage, and code duplication, to identify technical debt as soon as it is introduced.
- **Financial Measurement:**
    - FAGFDT converts technical metrics into financial values, calculating the potential cost of technical

  - debt in terms of maintenance, error correction, and impact on productivity.
  - It uses economic models to estimate long-term costs associated with different types of technical debt.
- **Reports and Interactive Dashboards:**
  - It provides detailed reports and interactive dashboards that present identified technical debt and its financial impact.
  - Reports highlight critical areas that require immediate attention and prioritize actions based on return on investment (ROI).
- **Strategic Planning:**
  - The tool allows managers and developers to plan technical debt resolution strategies based on cost-benefit analyses.
  - It integrates with project management tools to align technical debt mitigation efforts with development schedules and budgets.
- **Alerts and Notifications:**
  - It sends alerts and notifications to development teams whenever new technical debt elements are introduced into the code.
  - It suggests corrective actions and best practices to avoid the accumulation of new technical debt.

With the introduction of FAGFDT, the chaotic scenario would be reconfigured as follows:

- **Reduction of Technical Debt:**
  - Early identification and financial measurement of technical debt allow companies to address problems before they become critical, significantly reducing maintenance costs and system failure risks.
  - Companies would be able to keep their systems updated and secure, avoiding vulnerabilities and interruptions.

- **Increased Competitiveness:**
    - With the ability to quantify and manage technical debt efficiently, companies can allocate resources more effectively, promoting innovation and launching new products and services with greater efficiency.
    - Competitive advantage would be increased by the ability to continuously evolve software, improving customer satisfaction.
- **Sustainability and Growth:**
    - The data- and finance-based approach promotes sustainable development practices, allowing companies to better plan their technical debt mitigation strategies.
    - Companies could focus more on innovation and growth instead of being trapped in endless cycles of maintenance and problem correction.
- **Culture of Quality and Responsibility:**
    - The implementation of FAGFDT would encourage an organizational culture that values code quality and financial responsibility.
    - Developers and managers would be empowered with advanced tools that increase productivity and align their actions with the organization's financial objectives.

The introduction of a financial assessment and management tool for technical debt (FAGFDT) could transform a chaotic future scenario into a sustainable and competitive development environment. By identifying technical debt early and measuring it financially, this solution would ensure continuity, security, and evolution of software, allowing companies to thrive and adapt to future demands.

Other actions should be considered due to society's growing dependence on software systems [CISQ 2022]. It is recommended that organizations adopt rigorous quality standards, carefully evaluate third-party components, and integrate continuous technical debt correction into their development cycles. In addition, it is suggested to invest in improving the skills of professionals who make up the technical team and to raise awareness among project managers and executives about the relevance of technical debt. The adoption of these measures can help stabilize and reduce costs associated with poor software quality, ultimately promoting more sustainable and efficient companies.

5. Conclusion

Using speculative design to deeply explore the implications of current technological decisions for the future proves to be an excellent strategy for technical debt management. This practice helps visualize potential impacts and provokes critical debate about sustainability in software development.

In today’s world, where technology is the backbone of many products and services, technical debt is not just a technical problem but a strategic challenge that is reflected in organizational results. For many companies, especially those whose product or service is essentially software-based, managing technical debt effectively is crucial for maintaining product or service quality, operational efficiency, and market competitiveness. Companies that recognize and proactively address technical debt are better positioned to thrive in the digital era, ensuring that their systems remain robust and agile, ready to adapt quickly to market changes and innovations.

References

CISQ (2022). *The cost of poor software quality in the US: A 2022 report*. Available at: https://www.it-cisq.org/wp-content/uploads/sites/6/2022/11/CPSQ-Report-Nov-22-2.pdf.

Cunningham, W. (1992). The WyCash portfolio management system. In *Addendum to the Proceedings on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, volume 4, pages 29–30.

DiSalvo, C. and Lukens, J. (2013). The confluence of speculative design and community technology. *eScholarship*, University of California.

Dunne, A. and Raby, F. (2013). *Speculative Everything: Design, Fiction, and Social Dreaming*. MIT Press.

Kruchten, P., Nord, R. L., and Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. *IEEE Software*, 29(6):18–21.

Li, Z., Avgeriou, P., and Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, 101:193–220.

Loutfi, M. (2024). *Manual para prática de design especulativo*. Unpublished work.

Mankins, J. C. (1995). *Technology readiness levels: A white paper*. Available at: http://www.artemisinnovation.com/images/TRL_White_Paper_2004-Edited.pdf. Accessed: July 10, 2023.

McConnell, S. (2013). Managing technical debt (slides). In *Workshop on Managing Technical Debt* (part of ICSE 2013). IEEE.