Academe *Journal of Education & Psychology* *ISSN: 2249-040X*
Volume-16, (Special Issue: March 2026) *Peer Reviewed & Refereed Journal (IF: 6.25)*
PP: 93-99 *Journal Website:* www.academejournal.in

TECHNICAL DEBT -ITS VARIOUS ASPECTS AND ITS COUNTER MEASURES

Ravi Kumar Ramachandran

Cyber Security Practitioner and Researcher in Quantum Computing, AI & Data Science
Chennai, India

CISA, CISM, CGEIT, CRISC, CDPSE, CCSK, CISSP-ISSAP, CC, SSCP, CGRC, PMP, CIA, CRMA, CFE, FCMA, CIMA-Dip.MA, CFA, CEH, ECSA, CHFI, MS (Fin), MBA (IT), MBA (Ops), MBA(IB), COBIT-5 Implementer, Certified COBIT Assessor, ITIL 4 - Managing Professional, TOGAF 9 Certified, Certified SAFe5 Agilist, Professional Scrum Master and Product Owner -II, OCP-Cloud Architect, OCP-Gen.AI, CJUS(IITK),

CASE-Java

"Left unchecked technical debt will ensure that the only work that gets done is unplanned work"-Gene Kim

"Technical Debt is the coding you must do tomorrow because you took a shortcut in order to deliver the software today"-Katerina Trajchevska

"Some Debts are fun when you are acquiring them, but none are fun when you set about retiring them"-Ogden Nash

ABSTRACT

In this era of knowledge management and rapid technological transformation, the current industry is becoming increasingly dependent on technology and software development. Technical debt is an invisible and inevitable enemy every CTO needs to grapple with and bring it under control, else it could lead to huge losses. Though all technical debt is not detrimental, still it needs to be monitored considering the pace with which the technological advancements are happening. In this paper we will explore, the fundamental meaning of technical debt, its impact, the challenges in managing it and explore various countermeasures both technical and non-technical to bring it under control

KEYWORDS:

Technical debt, Architecture, AI, Machine Learning, Sunk Cost, Software development

INTRODUCTION

The topic of technical debt had become very important due to the imminence of software and software development in every enterprise. In addition, software applications serving various business needs have become very big leading to difficulties in maintaining it over a period of time. Software entropy or rot happens due to continuous changes done often ad hoc to software applications makes future changes more costly, difficult, time consuming resulting in technical debt. This is a drain on company revenues and profitability and the most dangerous part is, it is invisible. It is a nightmare to every CTO. Therefore, this topic is presented in this research paper detailing its various aspects.

STATEMENT OF THE PROBLEM

This Research study is undertaken to answer the following questions and it is answered number wise in the sections below.

1. What is a technical debt and what are the different types of technical debts?
2. How it is different from sunk cost?
3. What are the causes of technical debt?
4. What are the impacts of technical debt?
5. How to identify technical debts?
6. Do Machine Learning Systems have hidden technical debt?
7. What are the countermeasures to technical debt?

RESEARCH METHODOLOGY AND LITERATURE REVIEW

Since the selected topic is very complicated and evolving, an explorative research study is undertaken with the select review of literature to ascertain the answers for the aforesaid research questions and have been duly cited in the references section. The aim of this research study is an attempt to add to the existing body of knowledge in the technical debt management by adding different perspectives.

1. Meaning of Technical Debt and its Classifications

Ward Cunningham coined the term "technical debt" in 1992 to describe the long-term compounding cost of choosing fast "first-time code" over a better, more robust design. (Source: Wikipedia)

Technical debt happens when a software development team makes a change in the software application without considering the long-term impacts, but just to satisfy the current market demands and the internal stakeholders, which results in effecting future changes costlier or more difficult eventually leading to accumulation of more technical debt. In short, technical debt is the loss arising due to thinking and acting only in short term, without considering long term consequences.

A task needs to done. It needs be done right, once, and once and for all -meaning taking into effect all the long-term consequences. If it is not done that way, then a debt is created which is principal. For every future change, that principal -meaning that incomplete or erroneous task extracts extra cost, which is interest.

Once upon a time, there lived a king. He had a horse which participated in horse races and won him the first prize often! In one competition, it did not come first. So, the king got his horse checked by his veterinary doctor who prescribed a small bowl of medicine to be administered once a month over three months, with a promise that it will become the fastest horse in the earth! The king out of his greed, administered all the medicine overnight resulting in his horse becoming totally sick. The doctor, as a corrective measure, prescribed medicines to be given for next months. But alas! the horse after remaining sick and consuming medicine for three months, died. In addition, it needed a royal burial. Therefore, the act of wrong administration of medicine is the principal, and the consequent medical and burial expenses are the interest cost, whose total is the technical debt incurred by the king by his thoughtless action!

[1]Technical debt captures the extent to which design decisions that are expedient in the short-term can lead to increased system costs in future (Brown et al, 2010, Kruchen et al, 2012a)

Classification of Technical Debt

[1] Managing technical debt in software-reliant systems | Proceedings of the FSE/SDP workshop on Future of software engineering research

[2]According to Alves, there are so many different types of technical debt and its associated causes

Design Debt: Incurred due to the bad initial system design resulting in under-performance later

Code Debt: A code developed without coding standards leading to added maintenance later

Architecture Debt: Bad design or lack of modularity making adding features in future difficult

Infrastructure Debt: Insufficient or obsolete infrastructures making deployment difficult

Testing debt: Inadequate testing leading to flawed software

Documentation debt: No documentation or insufficient explanatory comments

Requirement debt: Insufficient or vague requirements resulting in a different unrequired product

People debt: Human errors due to lack of communication in the development team

2. Distinction between Sunk Cost and Technical Debt

The intent of this section elaborating the differences between sunk cost and technical debt is to enable Finance and Accounting Professionals to understand it closely to that they can assist in measuring and controlling technical debt.

- Sunk Cost is a management accounting term while technical debt is a technology term.
- Sunk Cost is past expenditure which normally would have happened in bulk or one time expenditure which is considered irrelevant for decision making. Technical Debt accumulates over a period by forced or thoughtless software or architectural changes misaligning from the original context of the system thus leading to suboptimal performance.
- Sunk Cost is mostly inevitable while technical debt though cannot be fully eliminated can be minimized and kept under control by continuous Governance.
- Sunk Cost is considered irrelevant in decision making, while technical debt plays a major role in ROI Calculations or upgradation to newer technology.
- Sunk Cost is visible and accounted while technical debt is invisible and manifests in system underperformance indicating poor technology governance including bad IT investment decisions which could be multifold leading to large scale retrenchments or even bankruptcy of the firm
- □ AI has no direct impact on sunk cost while it exacerbates technical debt by accelerating coding on a large scale without considering original architectural context.
- Sunk Cost comes under domain of project heads and CFO while technical debt comes under CTO.

[2] Identification and management of technical debt: A systematic mapping study - ScienceDirect

3. The Causes of Technical Debt

The following are the broad causes of Technical Debt, excluding those factors which are already covered under Classification of Technical Debt:

Thinking short-term: Deploying changes to software applications without proper change management process which does not consider all long-term effects and the overall architectural context.

[3]Insufficient Human Cognitive Capacity: Every software engineer has cognitive limits, while the software on which he is operating is huge and complex with so much of dependencies and links, which makes it practically impossible to consider the overall impact of the effected change.

[4]Laws of Software Evolution: M.M. Lehman in this paper, have stated that every software must be continually adapted as it evolves with the passage of time and changes made, else the degree of satisfaction provided by the software will decline with time. Also, as a program is evolved, its complexity increases unless work is done to maintain or reduce it. Since many organizations fail to pay attention to these laws of software evolution, technical debt happens.

[5]Fowler's Quadrant:

Technical debt quadrant has been given by Martin Fowler explaining the causes of technical debts. Reckless deliberate software development team just do not care. They do not have time to consider long term issues. Reckless inadvertent software development team do not know how to do a proper job without incurring technical debt. Prudent deliberate software development team know the long-term issues they are ignoring but decides to do it later. Prudent inadvertent software development realizes their mistake later and ascertains remedial actions to be implemented in future.

(Source: What is Technical Debt & How to Pay it Off, Examples [2025] • Asana)

4. THE IMPACTS OF TECHNICAL DEBT

Impact on business performance: Technical Debt will affect the business adversely not only in terms of profitability but also in terms of Goodwill and Customer satisfaction. Product not meeting customer expectations, cost of fixing constant

[3] Why Software Architecture Decays: The Cognitive Limits Behind Technical Debt and System Entropy | by Faisal Feroz | Medium
[4] Laws of software evolution revisited
[5] Technical Debt Quadrant

errors, bugs, fixes, patches will mount resulting in reduced revenues and due to downtime losses charged by the customer for the failure of software applications.

Impact on the Software Product: The product will falter on its functionality, reliability and resilience due to its buggy coding and flawed architecture. The cumulative effects of the changes and its associated technical debt will cause the product to be completely abandoned resulting in huge losses to the enterprise.

Impact on the employee morale: The software development team's morale will be worst hit as they will be spending their whole time on firefighting issues and fixing the errors. They would not have time to think and apply long term solutions. Also, due to constant employee turnover they will be lacking the context in which changes to be applied resulting in more technical debt. There would not be any Architecture Design Records (ADR) to apply solutioning in terms of the architectural context of the software resulting in more errors and loss of morale and employees leaving the organization.

5. IDENTIFICATION OF TECHNICAL DEBT

Some of the tools used to identify technical debt are discussed below:

[6]SonarQube is a fully automated platform for code inspection to detect bugs, security vulnerabilities, code maintainability issues, tracks code complexity and duplication to help ensure code quality and reduce technical debt.

[7]CodeClimate is a cloud-based platform used to analyze code quality and to identify improvement areas. It provides all the services like that of SonalQube and integrates with version control systems like GitHub and helps track technical debt.

[8]Squale is an open-source platform for managing technical debt and deals both in economic and technical aspects of quality

[9]Tech Debt tracker is a tool used by software engineering teams to identify, visualize and manage the accumulated "interest" on code shortcuts or outdated technology

[10]Jira is an effective tool for reducing technical debt by assisting in creation of dedicated issue types, utilizing customized dashboards, linking issues to Atlassian Compass which is an internal developer platform that helps team with centralized software catalogue to store all information about the software.

[11]AI Tools such as Natural Language Processing (NLP), Static Analysis, Deep learning, Cognitive Biasing can be used to identify technical debt.

6. HIDDEN TECHNICAL DEBT IN MACHINE LEARNING SYSTEMS

[12]D.Sculley and others have argued in their research paper, that though machine learning offers a fantastically powerful toolkit for building complex predictive systems quickly, these ML systems incur lots of technical debt, as in addition to the maintenance problems of traditional code, they have an additional set of ML specific-issues. This debt is difficult to detect as it exists at the system level, rather than at the

[6] SonarQube: Fight AI Slop & Verify AI Code | Sonar
[7] Code Climate - Powering your AI SDLC
[8] https://www.squale.org/
[9] Technical Debt Tracker Template | Miro
[10] Prioritize Jira Backlog and Reduce Tech Debt with Compass | Atlassian
[11] [2306.10194] Artificial Intelligence for Technical Debt Management in Software Development
[12] Hidden Technical Debt in Machine Learning Systems

code level. Traditional abstractions and boundaries may be corrupted since data influences ML system behaviour, and such traditional methods are insufficient to address this issue.

7. COUNTERMEASURES TO TECHNICAL DEBT

Technical Debt is a business Issue: Treating Technical debt as a business issue or the management issue than that of technology issue is the very first step an organization should take to tackle it effectively. Therefore, the entire organization should have a clear and common understanding of what is technical debt. [13]One organization defined it as a negative impact of technology on operational and technology costs, slower time to market and reduced flexibility.

Have a Dedicated Team: Once done with the identification of organizational level problem of technical debt and its common definition, a dedicated cross-functional team should be formed comprising software developers, architects, HR and Finance, CTO, and board members to work on it continuously with the adequately provided resources

Use AI-assisted Code Cautiously: AI agents often operate based on local prompt specific instructions rather than on overall-code context, resulting in duplication or recreating an existing functionality, or repeated codes, leading to bloated code-bases and excessive technical debt. Therefore, coders should treat AI generated code as code developed by another developer and should be audited thoroughly for a clean code.

Code Quality and Continuous Testing: Code developers should be encouraged to follow best coding practices and latest frameworks and continuous code reviews. No deployment before comprehensive testing should be the mantra.

Refactoring as a continuous routine: Refactoring as a process focuses on internal structure without changing the external behaviour. It focuses on software attributes of readability, maintainability, complexity reduction and reducing technical debt. It should be a continuous routine and be part of the organisation's culture.

Best Practices: Implement Continuous Integration and Continuous Delivery (CI/CD) practices. Continuous Integration ensures that changes are frequently merged into the main codebase and validated through automated tests. Continuous delivery ensures that software is deployable at any given time.

Agile Practices: Agile Methodologies such as Scrum, Kanban focuses on iterative development through sprint backlogs and frequent feedback loops. It breaks down work into smaller, manageable limits and delivers value incrementally reducing the risk of accumulated technical debt.

ADR and Clear documentation: Maintenance of Architectural Decision Records (ADRs) and clear documentation on Software design and codes, including API (Application programming Interface), README files, helps new developers understand the system thoroughly and reduces the creation of technical debt through misunderstanding.

Knowledge Organization: The Organization should foster a culture of continuous learning and upskilling so that the software development team is aware and well

[13] Tech debt: Reclaiming tech equity | McKinsey

trained in latest best practices and technology frameworks which will naturally lead to creation of software products with excellent quality and free of technical debts.

Management Accounting Metrics: Through management accounting metrics such as amount of unplanned work encountered during every planned and legitimate activity, by calculating hourly rates of software developers, idle time hours, system downtime hours, employee turnover ratio, difference between expected ROI and actual ROI, monetization of customer feedback known as Net Promoter Score (NPS) etc., technical debt can be quantified in numbers and remedial action can be taken division wise or at the enterprise level.

CONCLUSION

Technical debt, as a phenomenon is present everywhere in software. It always existed and will always exist, though it attracted so much of attention in the past few years, is since technology is used everywhere and the pace of its usage is increasing exponentially every day. Along with its usage, technical debt is also growing. In addition, with the advent of AI, this problem is getting aggravated, as it can scale up the coding without context, and can create massive technical debt. Since technical debt can make a company bankrupt, now it has become board level priority and scrutiny. We have examined various aspects of technical debt and proposed counter measures which though it cannot eliminate the problem totally, but it can bring it under control and usher in implementation of industry best practices.

REFERENCES

1. Managing technical debt in software-reliant systems | Proceedings of the FSE/SDP workshop on Future of software engineering research
2. Identification and management of technical debt: A systematic mapping study - ScienceDirect
3. Why Software Architecture Decays: The Cognitive Limits Behind Technical Debt and System Entropy | by Faisal Feroz | Medium
4. Laws of software evolution revisited by M.M.Lehman
5. Technical Debt Quadrant by Martin Fowler -martinfowler.com
6. SonarQube: Fight AI Slop & Verify AI Code | Sonar
7. Code Climate - Powering your AI SDLC
8. https://www.squale.org/
9. Technical Debt Tracker Template | Miro
10. Prioritize Jira Backlog and Reduce Tech Debt with Compass | Atlassian
11. [2306.10194] Artificial Intelligence for Technical Debt Management in Software Development
12. Hidden Technical Debt in Machine Learning Systems
13. Tech debt: Reclaiming tech equity | McKinsey