Grenze International Journal of Engineering and Technology, January Issue

Navigating Technical Debt in Agile Practices: Insights from the Indian Context and Strategies for Effective Management

Mitali Chugh[1] and Anushtha Vishwakarma[2]
[1]UPES, Dehradun, India
Email: mitalichugh11@gmail.com
[2]Senior Research Associate, Research Nester, Noida, India

***Abstract*— Technical debt has become a major issue in software engineering due to the conflict between getting new software solutions to the market quickly and attaining the strategic benefits of modern software architectures. It comprises defects existing at the architectural level of a system and which make its operations and maintenance costly. This study aims to understand the Indian context of TD in an agile environment and uses a survey as the data collection instrument. Hence, this work looks at how agility impacts TD growth, reveals practices that either aggravate or reduce TD, and gives examples of TD management. Further, it reviews the issues that organizations face in managing TD, besides identifying the lessons that can be learned from case study successes and failures. As such, this study provides insights into Agile and technical debt, as well as suggestions for organizational improvements, and implications for future leadership for governments and industry bodies regarding the enhancement of SD on an international level.**

***Index Terms*— Technical Debt, technical debt management, technical debt repayment, TD causes, TD effects.**

I. INTRODUCTION

The term "technical debt" (TD) was first coined by Ward Cunningham, one of the au- thors of the Agile Manifesto [1]. He likened issues with code to financial debt, stating that it is acceptable to borrow against the future if it is paid off. The term was used by Cunningham while developing a financial application in Smalltalk. The need for technical debt arises from the necessity of code refactoring and improvement, as code that is not well-suited leads to problems with structure, architecture, duplication, potential bugs, test coverage, documentation, and more. These issues result in technical debt, negatively impacting code productivity. Common types of technical debt include architecture debt, code debt, defect debt, design debt, documentation debt, infrastructure debt, process debt, requirement debt, and test debt.

Organizations often face a tradeoff between quality and speed due to factors such as imperfect coding skills, inconsistent business decisions, time and budget constraints, inadequate design selection, or a strategic choice to assess market fit. If the technical debt remains unaddressed, it can lead to symptoms such as concerns about code quality, longer time to market and release cycles, adverse user experience, reduced team agility and productivity, product bugs, and challenges with scaling and implementing new technologies.

Grenze ID: 01.GIJET.12.1.522_1

While software development teams may take shortcuts to deliver software earlier, not all shortcuts are detrimental. But if the team is concerned about TD incurred and energetically repays it, or at least a part of it, they can alleviate its effects. Software development teams can opt for many pathways to address TD, as there is no one-size-fits-all solution. To decide the suitable approach, the status quo should be carefully understood and evaluated. This can be accomplished through project management, where the impact of TD is measured, and a project plan addresses the effects of TD. Even though diverse situations require different approaches to address the impact of technical debt, the overarching rule should always be to avoid technical debt whenever possible.
The present study aims to provide insights into technical debt and the best practices for its management. The following research questions have been explored:

- In what way do the years of experience of software developers determine the views they have on the causes and effects of technical debt in software development projects?
- What are the most important technical debt management practices from this perspective?
- Perspectives of software developers and the frequency of usage of these practices in real projects?
- What is the attitude of the software developers toward the connection between agile software development methodologies and technical debt management? Which specific agile practices lead to the addition of technical debt, and which of them help in its reduction?

The organization of this paper is structured as follows: Section 2 presents a survey of the literature on technical debt to establish the background for the research study. The methodology adopted in this research is described in section 3 concerning the method used in collecting/analyzing data. Section 4 presents the findings, where specific information about the views of Indian software developers toward technical debt is provided. Lastly, in Section 5, the main findings of the paper are discussed along with some recommendations for future research and practical implementation of the suggested model for practitioners.

II. Literature Review

TD is a widely recognized concept in software development and start-ups, and numerous researchers have conducted studies on its management, prioritization, and repayment. In this section, we present related studies that provide an overview of the TD landscape.
Besker et al. (2018) focused on TD in the context of startups and found that factors such as the startup phase, developer experience, founders' software knowledge, and employee growth influence the intentional accumulation of technical debt [2]. Klotins et al. (2018) studied the prevalence of TD practices, factors, and outcomes in start-ups [3]. Verdecchia et al. (2018) examined architectural TD, including its identification, classification, evaluation, characteristics, publication trends, and potential for industrial adoption[4].
Gilson et al. (2018) emphasized the significance of engineering practices in managing TD. Their study investigated quality-related processes in Scrum development teams and found that even junior developers prioritize quality indicators despite having less knowledge about them [5]. Ren et al. (2019) focused on admitted technical debt (SATD), a form of TD intentionally introduced during development. They used Convolutional Neural Networks (CNN) to classify code comments as SATD or non-SATD, conducting trials with a large collection of code comments and validating their method's effectiveness for SATD classification [6].
Spinola et al. (2019) compared automated and human-centered TD identification and concluded that automation alone, using tools, is insufficient for complete TD identification, highlighting the need for human intervention to achieve more accurate results [7]. Ampatzoglou et al. (2020) conducted an empirical study on TD principles and interests. Their findings demonstrated that higher levels of TD principals are often associated with higher levels of interest, emphasizing the importance of prioritizing specific quality attributes in refactoring opportunities[8].
Iammarino et al. (2021) explored the relationship between code refactoring and SATD. Their study showed that refactoring is more likely to coincide with SATD deletions compared to other commits, often implemented alongside other quality improvement actions [9]. Siavvas et al. (2022) investigated the capability of common TD markers to identify security risks in software products using a machine-learning approach. Their results suggest that TD markers can serve as indicators of security risks [10].
Ramac et al. (2022) provided empirical insights into the understanding and use of TD concepts. They analyzed causes and effects and found that time pressure or deadlines were the most frequently cited causes of TD, while delivery delay, low maintainability, and rework were the most mentioned effects[11]. Besker et al. (2022) explored the use of incentives for TD management and found that implementing a TD manage- ment strategy, particularly one centered on incentives, can significantly impact the extent of TD in software[12].
A survey-based study was conducted by Gilberto et al. (2024) with 53 participants, who worked in the AI field and practices; the questions were asked to collect data regarding the occurrence of AI technical debt problems in

both code and architecture dimensions and their severity. The work also aimed at revealing the approaches that practitioners use to check for and remove such problems. Important observations include the effects of AI technical debt on the quality of AI-based systems, such as the extremely negative effects of undeclared consumers on security, and how the jumbled model architecture increases code complexity and hence maintainability. As observed, there are several caveats imposed by these restrictions. Currently, there are only a few tools supporting practitioners, and most of the time, practitioners only apply manual identification and refactoring efforts. The study was conducted to increase the aware- ness of researchers for detecting and managing AI technical debt[13].

Stochel et. al. introduce TD management using the Continuous Debt Valuation Approach (CoDVA), and the findings show the effectiveness of using this approach. Overall, the business value perceived from technical debt refactoring went up by 27%, engineering velocity was up 39%, engineering predictability was up by 60% and lastly, it was observed that the effort required towards product release stabilization was reduced by 50%. All in all, in 86 percent of cases, the satisfaction of developers improved or remained the same, or was at least not worse. A vast majority of developers reported that the chosen approach to technical debt management helped them make technical decisions, in creating new functionality and enhancing developers' experience while interacting with the product code[14].

III. METHODOLOGY

A. Study Design and Participants

This research uses a cross-sectional survey questionnaire to understand the stance and handling of technical debt among software developers. The questions are divided into numerical and non-numerical, which will enable the creation of a diverse picture of how developers without regard to experience and background practice TD, and how they understand it within software development projects. Since this is a pilot study and since the specific areas of specialization of the participants were not restricted, the participants in this study include software developers who have one or more years of experience in the industry, but not necessarily more than five years of experience. This is why the survey aims to reach out to as many respondents as possible so that a rich picture of technical debt can be obtained. The collection is done over the internet, social networking sites, and industry groups and associations to capture as diverse a sample as possible. This eliminated the possibility of participants giving insights that are alien to software development, hence reducing technical debt significantly.

B. Survey Instrument

The survey instrument consists of multiple sections designed to capture different aspects of technical debt:

The demographic information section gathers simple and universal user parameters, the values of which are age, gender, and experience in software development. It assists in understanding ahead of time how the reactions may be affected by various demographic attributes to draw patterns. Next is the familiarity with Technical Debt section, which first asked respondents if they have any understanding of technical debt before going on to the next questions, since the survey questions depend on that factor.

The causes of Technical Debt section employs a 5-point Likert scale with response options of Strongly Agree, Agree, Neutral, Disagree, and Strongly Disagree to measure the respondents' perception of some of the forces that would lead to having technical debt, namely, Deadlines, inadequate planning, lack of knowledge or expertise, and poorly defined processes. In the effects of TD, the respondents answer the question on the frequency of perceiving some of the impacts of TD, such as the low quality of software products, delays in delivery, low maintainability, and higher effort for the rework, all Frequent, Occasionally, Rarely, or Never.

The technical Debt Management Practices section aims to identify to what degree various kinds of technical debt management are applied in the projects in which the respondents are involved; for this purpose, a frequency scale was offered (Always, Often, Sometimes, Rarely, Never). In Agile Practices and Technical Debt, the respondents give their views regarding the association of agile methodologies with the management of technical debts, given the options that represent different stances. As for the third objective, the Experience and Perception of Technical Debt section describes how participants' experience affects their views on technical debt and its consequences. Global Perspectives on Technical Debt Management section, the respondents are asked about their experiences concerning the effect of culture and region on the management of technical debt. The last question in the survey is an open question that allows the respondents to give further opinions or share comments about technical debt within software development.

As for the survey, numerical data are described and analyzed using descriptive as well as inferential statistics. Measurement statistics accurately describe the demographic data and the summary of the answers obtained. In

analyzing the results obtained from the open-ended questions, thematic analysis of data is used. The identified common threads and words are to gain a better understanding of the respondents' experience and recommendations concerning the management of technical debt. The themes and keywords, based on the most often used words, are shown in a word cloud.

IV. Results and Discussion

A. Demographic Information

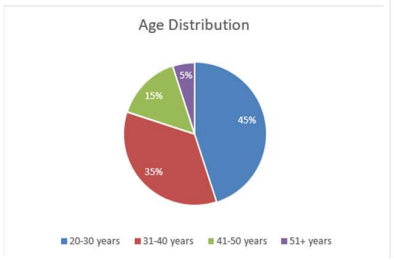

Fig 1

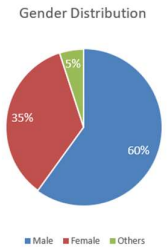

Fig 2

B. Years of Experience Distribution

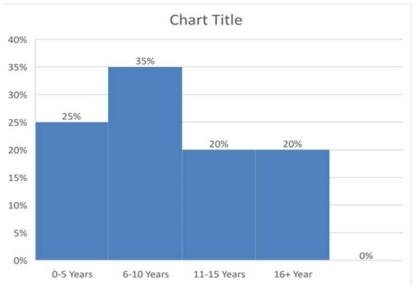

Fig 3

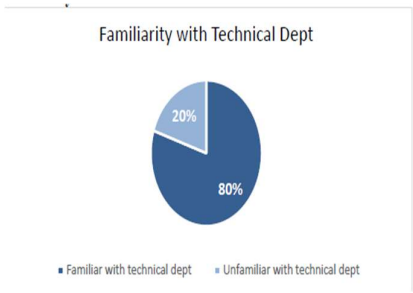

Fig 4

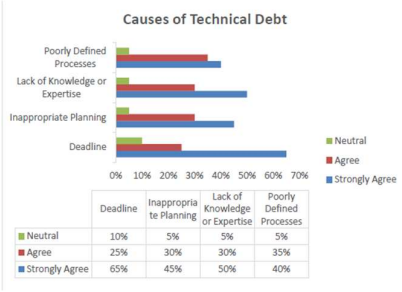

Fig 5

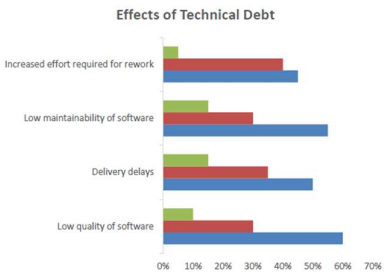

Fig 6

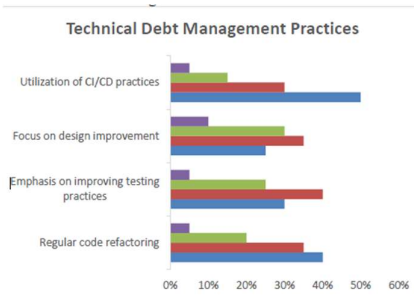

Fig 7

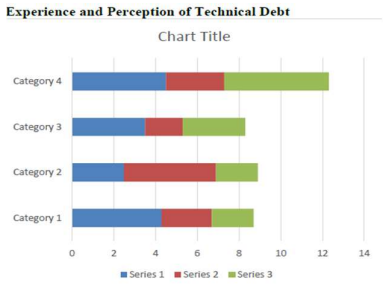

Fig 8

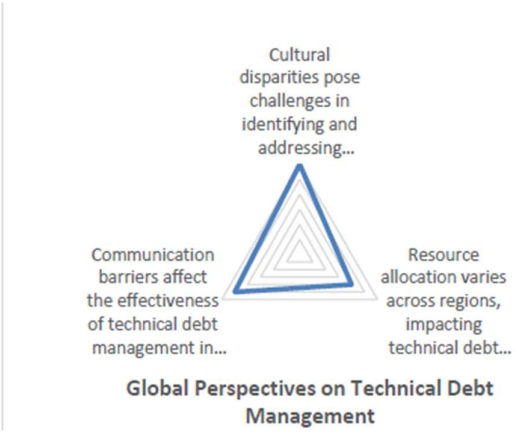

Fig 9

V. General Feedback

These empirical results provide insights into the causes, effects, and management practices of technical debt in software development, as well as perceptions across different demographics and regions.

Comparing the survey with the received results, it became clear that the causes of technical debt and the corresponding effects can be viewed quite differently depending on the years of experience of the respondents in software development. Technical debt is defined by professional developers as tactical and anthropomorphic, meaning that insufficient planning and communication cause it. This indicates that staff superiors with professional experience understand the necessity of integrated project administration and the relationship of organizational activities to technical debt. On the other hand, novices are more inclined to view technical debt in strictly technical terms, such as code being old or not having sufficient testing conducted on it. This is a more parochial view of technical debt. Such differences call for special training and awareness-raising initiatives that would encompass both the big picture and the details of technical debt management to work in parallel with employees' experience levels.

The findings further support that code refactoring and CI/CD are considered the most efficient techniques for addressing TD. Participants feel that they have gone a long way in helping to eliminate technical debt because they help maintain the quality of the code, as well as help in the early identification of potential problems. However, it seems that the frequency with which these practices are set up differs, and numerous respondents mentioned time and resource limitation issues. This implies there is a disconnect between the identification and utilization of effective strategies in the recruitment and selection of employees. These barriers could be overcome by improving the distribution of resources for the implementation of technical debt management in projects, thus improving project planning. Moreover, the results suggest that test attitudes regarding the efficiency of certain practices and the extent to which they are utilized in various projects, especially since big projects that incorporate complex elements such as regular refactoring and CI/CD, are more apt to embrace such approaches.

Thus, the respondents have a rather divided opinion on the extent to which agile methodologies can help manage technical debt. Others think that agile practices do a good job of managing the TD issue through iterative, incremental, and cooperative work cycles and built-in testing and feedback mechanisms. Nevertheless, some people believe that agile increases the amount of technical debt since it is rather fast. This division leads to the conclusion that although some approaches and methodologies can be used for technical debt management, the issue here is in the ability of organizations and companies to implement a series of agile principles. Some of the practices to recommend include sprint planning, continuous integration as well and code reviews. The discussion highlights the importance of having technical debt as the subject of concern in agile teams' work, as well as addressing that having speed in work means incurring technical debt without neglecting its consequences.

Some cultures and regions play a major role when influencing the management of technical debts. In this survey, it has been established that certain cultures continue to favor speedy delivery / working towards temporary objectives that can beget high technical debt, as compared to other cultures that value the importance of the practice in maintaining sustainable technical debts. The main idea The following indicates that managing technical debts across various cross-functional teams fosters cultural variance, which hinders reinforcing best

practices of technical debts across international groups. Also, the decentralized distribution of resources hinders the efficient control of technical debt across the regions. For example, it is easier for teams in the newsrooms to follow best practices because they have access to tools and training materials that their counterparts in resource-poor settings lack. Another issue that exacerbates technical debt in global teams is communication breakdown, because it hinders cooperation and fosters miscommunication in priority settings. To address the mentioned problems, cross-cultural training should be encouraged, resources should be procured in all the regions, and the organization should strengthen the communication systems to support collaborative work in an international team.

VI. Implications for Research And Practice

Diverse Perspective on the Technical Dept: The subject matter of technical debt is viewed in different ways depending on the amount of experience a developer has. The reason why there is a need to dig deeper into how different demographics of developers view and handle technical debt. Further research should be undertaken for the precise identification of the factors related to the above perceptions and different training programs that may be required because of these patterns.

Effectiveness of Management Practices: The identification of good technical practices to manage this technical debt, refactoring, Continuous Integration/Continuous Deployment, raises questions regarding the conditions and environment that favor most of these technical debt management practices. More specifically, future research should inspect the organizational, project, and cultural enablers and barriers to the application of these practices.

Agile Methodologies: Such research call indicates that although there are positive approaches towards agile methodologies in managing technical debt, there are equally negative opinions about it, thus the need for more research on determining the impact of agile practices on technical debt. Research should be aimed at defining which of the adopted practices affect technical debt and how agile teams can maintain speed while having quality in code.

Global Perspectives: The review of cultural and regional influences on technical debt management indicates the necessity of cross-cultural research. Technical debt practices require more research on how various cultural frameworks and resource limitations affect the implementation of TD and employable concepts for managing it in international settings.

A. Implications for Practice

- Experience-Based Training: It is recommended that organizations establish training sessions that cover both the tactical and operational angles of TD management and that suit entry-level technical employees. While some employees may require strategic planning and senior project management training due to their years of experience, others may need technical training, most basic to developers.
- Adoption of Effective Practices: Considering the positive impressions of refactoring and CI/CD practices' frequency, these practices should be valued within the development cycles. Some of the barriers to using HSS include limited time and lack of resources, among others; if these barriers are removed, HSS usage will increase. For these, they call for investing in automation tools and incorporating the practices of technical debt management into project plans and schedules.
- Optimizing Agile Practices: Thus, it is confident that technical debt should be managed by agile teams, and the principles of its management should be incorporated into the framework of the agile approach. Some of the best approaches one can adopt include sprint planning, continuous integration, and code reviews, which, if adopted, help reduce the issue of technical debt. Educating the agile teams on the need to combine the speed of development with the quality of code will improve the efficiency of the agile processes regarding technical debt.
- Cross-Cultural Collaboration: One can consider cross-cultural education and adjust resource distribution based on region, depending on the organization's strategies for managing technical debt. It will be possible thereby to standardize the metrics and toolkits for the management of technical debt in the organization and enhance the cooperation of multidisciplinary teams through the correct use of communication tools.
- Continuous Feedback and Improvement: As can be seen from respondents' general feedback concerning the situation, technical debt management needs to be updated and improved constantly. It is suggested that organizations should periodically collect feedback from development teams, leaders, and stakeholders, and study common problems that need to be addressed for improved technical debt management. Thus, continuing education, enhancement of tooling, and the cohesiveness of the teams are vital steps in this process.

VIII. Policy Recommendations for Governments and Industry Bodies

- Standardized Metrics: Encourage the development of standardized reporting procedures and indicators for technical debt. To enable benchmarking and comparison, common measurement frameworks can be established by governments and industry organizations.
- Incentives for Training: Give organizations incentives or subsidies for spending money on workforce education and training programs that follow agile. This might encourage organizations to adopt the best practices.
- Support Research and Best Practices: Fund initiatives that investigate the relationship between agile development methods and technical debt. Encourage the distribution of standards of excellence through industry conferences and publications.
- Regulatory Frameworks: Establish legal structures that promote openness in software development procedures, particularly regarding technical debt. Encourage organizations to make their technical debt levels and management techniques accessible in their annual reports.
- Collaborative Forums: Create collaborative forums or industry alliances where organizations can exchange knowledge and best practices about the adoption of agile software development and technical debt management.
- Government Procurement Guidelines: Create regulations for government procurement that consider the ability of an organization to handle technical debt. When granting contracts for software development projects, take technical debt measures into account during the evaluation process. By implementing these suggestions into practice, organizations may improve their agile processes and efficiently manage technical debt, and organizations in both the public and private sectors can promote an environment that encourages transparency, knowledge sharing, and best practices in the field of software development. Agile methodology is a consistent technique where changes can be made at each stage to guarantee the task will accomplish its goals [15]. The findings of this research will be useful in understanding technical debt and its perception and management among software developers across organizations in India. The results reveal that there are distinctive differences concerning the meaning of technical debt depending on the years of experience as developers, where senior developers take into consideration both the strategic and human elements [16], while junior developers are aware of the technical aspects. In this case, the cases of commonly recommended practices like refactoring as often as possible and practicing CI/CD show that such practices help to address technical debt, although there remains the challenge of implementing these solutions [17]. Concerning India, it is crucial to focus on some social and geographical aspects when speaking about technical debt. Resource constraints and communication barriers are identified as important challenges. These results raise questions regarding organizations' commitment to cross-cultural training, adequate funding, and the effectiveness of their means of communication when it comes to managing technical debt. Also, the variable perceptions concerning the agility approaches show that the effect is dependent on the approach used in executing the idea and practice of agile. Thus, the organizations operating in the context of India should focus on the need for specific staff training, efficient resource planning, as well as management of improvement processes. Considering the obtained results, this study underlines the need to focus on the context for better management of TD in software development projects conducted in India.

VIII. Conclusion

Considering the obtained results, this study underlines the need to focus on the context for better management of TD in software development projects conducted in India. The results of this study point to several intriguing directions for future research as the software development landscape keeps evolving. These consist of:

- Agile Practice Evolution: Examining whether agile techniques change over time and adapt to shifting technological environments, including the inclusion of cutting-edge approaches like DevOps and DevSecOps.
- Global Agile Adoption Trends: Investigating shifting patterns in the adoption of agile techniques across various sectors and industries, as well as how they affect global technical debt.
- Advanced Technical Debt Metrics: Developing more sophisticated standards and tools for evaluating technical debt, such as predictive models that may foresee the effects of debt on software quality and project schedules.

- Cross-Cultural Agile Practices: This involves an investigation of how cultural differences affect technical debt management and agile techniques, with a particular emphasis on methods for bridging cultural gaps among global software development teams.
- Regulatory Frameworks: Examining how well regulatory frameworks and guidelines function to reduce technical debt and encourage openness in software development procedures.

In conclusion, the present research provides significant novel perspectives on the intricate interaction between agile development practices and technical debt in the field of software development. Organizations
are given practical ideas to improve their agile practices and technical debt management by synthesizing a global viewpoint. Further research in this domain has the potential to advance the discipline even further and ensure that software development is agile, adaptable, and effective in confronting a variety of evolving challenges.

REFERENCES

[1] K. Schwaber and M. Beedle, Agile Software Development with Scrum. Upper Saddle River, NJ, USA: Prentice Hall, 2002.

[2] T. Besker, A. Martini, R. E. Lokuge, K. Blincoe, and J. Bosch, "Embracing technical debt, from a startup company perspective," in Proceedings of the 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2018, pp. 415–425, doi: 10.1109/ICSME.2018.00051.

[3] E. Klotins et al., "Exploration of technical debt in start-ups," in ICSE-SEIP '18: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, 2018, pp. 75–84.

[4] R. Verdecchia, I. Malavolta, and P. Lago, "Architectural technical debt identification: The research landscape," in Proceedings of the 2018 International Conference on Software Engineering (ICSE), 2018, pp. 11–20, doi: 10.1145/3194164.3194176.

[5] F. Gilson, M. Morales-Trujillo, and M. Mathews, "How junior developers deal with their technical debt?," in TechDebt '20: Proceedings of the 3rd International Conference on Technical Debt, 2020, pp. 51–61.

[6] X. Ren, Z. Xia, D. Wang, and J. Claims, "Neural network-based detection of self-admitted technical debt: From performance to explainability," ACM Transactions on Software Engineering and Methodology, vol. 28, no. 3, pp. 1–45, 2019.

[7] R. O. Spínola, N. Zazworka, A. Vetro, F. Shull, and C. Seaman, "Understanding automated and human-based technical debt identification approaches: A two-phase study," Journal of the Brazilian Computer Society, vol. 25, no. 1, 2019, doi: 10.1186/s13173-019-0087-5.

[8] A. Ampatzoglou et al., "Exploring the relation between technical debt principal and interest: An empirical approach," Information and Software Technology, vol. 128, p. 106391, Aug. 2020, doi: 10.1016/j.infsof.2020.106391.

[9] M. Iammarino, F. Zampetti, L. Aversano, and M. Di Penta, "An empirical study on the co-occurrence between refactoring actions and self-admitted technical debt removal," Journal of Systems and Software, vol. 178, p. 110976, 2021, doi: 10.1016/j.jss.2021.110976.

[10] M. Siavvas, D. Tsoukalas, M. Jankovic, D. Kehagias, and D. Tzovaras, "Technical debt as an indicator of software security risk: A machine learning approach for software development enterprises," Enterprise Information Systems, vol. 16, no. 5, 2022, doi: 10.1080/17517575.2020.1824017.

[11] R. Ramač et al., "Prevalence, common causes and effects of technical debt: Results from a family of surveys with the IT industry," Journal of Systems and Software, vol. 184, p. 111114, 2022, doi: 10.1016/j.jss.2021.111114.

[12] T. Besker, A. Martini, and J. Bosch, "The use of incentives to promote technical debt management," Information and Software Technology, vol. 142, no. Oct. 2021, pp. 106–740, 2022, doi: 10.1016/j.infsof.2021.106740.

[13] G. Recupito et al., "Technical debt in AI-enabled systems: On the prevalence, severity, impact, and management strategies for code and architecture," Journal of Systems and Software, vol. 216, p. 112151, May 2024, doi: 10.1016/j.jss.2024.112151.

[14] M. G. Stochel, T. Borek, M. R. Wawrowski, and P. Chołda, "Business-driven technical debt management using Continuous Debt Valuation Approach (CoDVA)," Information and Software Technology, vol. 164, p. 107333, Apr. 2023, doi: 10.1016/j.infsof.2023.107333.

[15] S. Q. Mir and S. M. K. Quadri, "Component-based metric for evaluating availability of an information system: an empirical evaluation," Int. J. Inf. Technol., vol. 11, no. 2, pp. 277–285, 2019, doi: 10.1007/s41870-018-0220-2.

[16] P. Agrawal, S. Goyal, and A. Jandwani, "A novel agile based framework for employee promotion," Int. J. Inf. Technol., pp. 4481–4487, 2024, doi: 10.1007/s41870-024-02071-x.

[17] S. Vaidyanathan, C. Arumugam, K. Jaganathan, S. Ramesh, A. Anand, and K. R. Anandan, "LeSS agile projects: a machine learning driven empirical model to predict the human resource allocation," Int. J. Inf. Technol., vol. 16, no. 2, pp. 861–870, 2024, doi: 10.1007/s41870-023-01513-2