International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.4 (2025) pp. 7599-7605
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Automated Technical Debt Assessment In Legacy Banking Applications

Rajesh Kumar*

4236 Balandre ln, Mckinney , TX , 75070
* **Corresponding Author Email:** rajesh11985@gmail.com - **ORCID:** 0009-0001-6485-7885

Article Info:

DOI:10.22399/ijcesen.4065
Received : 20 August 2025
Accepted : 06 October 2025

Keywords

Technical debt,
automated testing,
old banking applications,
software maintenance,
static analysis,
software quality

Abstract:

When it comes to banking operations, traditional architecture and obsolete software systems are by no means extraneous. However, the introduction of such systems does carry with itself the burden of technical debt, since it inhibits the ease of change, invites operational risk, and makes the raising of the cost of maintaining such a system inevitable. Making use of technical debt automation is a reasonable approach that is remarkably helpful in identifying, measuring, and prioritizing code and architecture deficiencies in such legacy systems in a structured way. In this article, we explore the methods and techniques aimed at the detection of technical debt in bank-related systems with a focus on software analysis, particularly static analysis, and architectural metrics, and consider regimes of their estimation incorporating machine learning algorithms. In addition, the study addresses certain other characteristics that are application-dependent including issues of compliance, concerns about the availability of services, and compliance with modern digital technologies. The findings turn the attention toward the advantages of introducing automated evaluation tools to simplify the appreciation process, reduce the exposure of the bank to risk and enhance the prospects of updating bank software environments. Monolithic architecture and legacy banking applications, typically based on old technologies, remain significant to financial institutions. However, the operation and improvement of such systems increase the technical debt that damages changes, imposes operational risk, and raises maintenance costs. To combat this, there is an analysis of which is quite systematic – automated technical debt analysis, which is designed to find, measure and prioritize coding and architectural problems in such systems. The article studies and facilities of the automated identification of technical debt in banking software development applies, focusing mainly on how these can be achieved through software static analysis, architectural metrics, and prediction models that rely on artificial intelligence. Furthermore, some defects investigate technology dependence compliance, antifraud, and satisfaction from digital area. The findings further emphasize the significance of automated assessment systems to speed up the making of decisions process, reduce risk, and smoothen the process of transformation.

1. Introduction

Technical debt is an acceptance in software development which is when the solution constructed is done in such a way which enhances the ability in meeting its short-term plans but without considering an increase in costs and risks in the future. Address the concept of technical debt, in other terms, refers to measures taken earlier in its development stages to save time or money over time, which if not done, is likely to have negative implications like poor quality software hindering maintenance and affecting innovation speed. Dealing with technical debt to facilitate the development of new functional and adaptive features has become a cultural trait in every software development project that hopes to see some growth happening.

It can be said that systems and applications in the banking sector have unique characteristics. The reason is that a significant portion of the financial sector is still using systems that have been in operation for several scores. These traditional applications can be deployed to manage client

account details, prepare regulatory reports, perform operations such as transaction processing, among others. While these systems are quite stable, their foundation is outdated, and they have capabilities that are concerned with monolithic and inapposite solutions for modern day business requirements. It is in this regard that there is growing anxiety about the existence of such systems from a technical debt perspective, and this diminishes the ability to enhance the system, thus promoting high costs in maintenance, liability and other business risks that affect the bank.

According to Juan Carlos Soto, how do companies typically evaluate their level of outstanding debt in terms of compute and datacenter resources in usage fitting? Acquisition costs are only one type of debt. Startup culture and spending fast is a debt as well. There are also existing definitions like tech debt, which is just another type of debt. What he means is that given all those kinds of debt, the old IT processes of investments might not be the solution. Do you agree that data collection has taken years to be of prime importance over other issues such as software design costs and telecommunication expenses?

2. What is the Technical Debt of Legacy Banking Systems?

The following are the characteristics of Legacy Banking Applications.

1.Traditional banking systems are monolithic syste ms, normally huge and developed
in decades for very important financial operations. Most systems are implemented in older languages li ke COBOL, PL/SQL, or an old version of Java and executed on a mainframe or infrastructure being a big conglomerate of incubation environment. Thesy stems have proven to be reliable and robust; howev er, changes are difficult to make as they are enormo usly complex and not modular. Besides, legacy ban k systems are usually installed and interfaced with a whole lot of external services through hard-
coded interfaces, which create a culture of high inte rdependency and low flexibility in adopting new di gital technologies.

2. Typical Originators of Technical Debt.

Technical debt for legacy programs can be attributed to the following reasons or causative factors:

Obsolescent technologies: Obsolete languages, frameworks, and platforms provide limits to maintainability and availability of skilled professionals.

Rigid architecture: A monolithic architecture generally does not entail modular upgrades that could be integrated with modern cloud-native and API-driven systems.

Lack of documentation: Critical design and implementation knowledge get lost over the course of time, leaving behind an implied indebtedness of some kind where there is a dependence on the shrinking human resources.

Workaround solutions and temporary patches: Due to the ever-changing regulations and market forces, it is always a temporary fix, which in the long term will be considered a liability.

Indentured coding practices: Different teams working with variation in the standards of development, along with several decades of modification, adding to the code duplication, redundant value, and lesser readability of parameters considered.

3. Issues of managing technical debt manually.

There are a lot of technical debt management difficulties in legacy banking applications that need to be administered manually. Firstly, the size of these systems, counting in millions of lines of code, defies analysis without tool assistance. Secondly, a lack of documentation and transparency prohibits human perception of dependencies and existence of risks. Thirdly, such evaluations are time-intensive processes, and inconsistent; some may also be prone to human errors in certain cases while others run the risk of missing out on significant errors or inefficiencies. Lastly, in the banking environment where regulations are subject to change, manual evaluation techniques cannot keep pace, in order to sufficiently and accurately allow compliance and operational robustness.

3. Automated Technical Debt Assessment.

The scoped and defined automated assessment.

Technically, automated assessment of technical debt indicates the use of software tools, algorithms, and computing methods to systematically identify, measure, and track technical debt signs present in software systems. On the other hand, such an automated assessment allows an objective, reproducible, and scalable appraisal of the review as opposed to manual reviews where the reviewer applies his/her judgment and expertise in carrying out the review. The automated assessment will then analyze many facets of the debt, including issues at the code level, architectural waste, documentation, and dependency risks, and present a single interface from the perspective of whether the system is maintained and exploited well.

Tools and Techniques
Technical-Debt analysis is automated by means of the given tools and techniques:

Static Code Analysis: Static code analysis tools such as SonarQube, CAST Highlight, and PMD scan the source code for detection of code smells, duplications, complexity, and standard code violations.

Metrics-Based Approaches: Software quality metrics (e.g., cyclomatic complexity, maintainability index, coupling and cohesion measures) represent a quantitative measure of structural weaknesses and are used as indicators of change in their status over periods of time.

Machine-Learning Models: Newest tools analyze previous defects and historic patterns of software evolution to hint at areas of technical debt along with refactoring priorities.

Architecture Analysis Tools: The parallel architecture recovery tools bring to light architectural breaches of set design principles, stratified architecture, and dependency circles in massive banking programs.

Debt Visualization Dashboards: Visualization frameworks supply stakeholders with an interactive dashboard that reveals debt hotspots and trends.

Benefits of Automation
Correctly measuring technical debt may provide immense value in the management of bad old banking systems:

Productivity: Automated tools, in contrast to manual testing, can review millions of lines of computer code in a fraction of a second.

Accuracy: Being rule-based, objective tests reduce human factors while still providing homogeneous results in large teams and projects.

Scalability: As systems are required to be practically 24-in-the-day in monitoring systems, the automated approach is applicable for large-scale, mission-critical banking systems.

Early detection: Automated systems are able to detect risks at early stages when these do not yet grow into big failures or compliance issues.

Decision Support: Such data-driven insights help to make clear the information managers should focus on with respect to debt repayment strategies and in establishing resources, aligning modernization efforts with business intents.

4. Implementation Challenges

Connectivity to Existing Banking infrastructure.
These banking systems are a vital part of the basic financial system, payment system, and regulatory systems. With disparate technologies, proprietary systems, and loosely coupled architectures, it becomes challenging to get any automated technical debt assessment tool into these areas. In many cases, these automatic tools may have to be run with access to either source code or build pipelines or runtime data, all of which may not be readily available or compatible with the older infrastructures. One major implementation challenge, therefore, is to ensure that such seamless integration does not interfere with the core banking activities.

Data Privacy and Security.
Financial and personal information is highly sensitive; therefore, stringent regulations, including GDPR, PCI DSS, and local banking regulations, have been laid down for banking systems. Most automated evaluation tools require broad access to system logs, code, and system configuration files. These may potentially contain sensitive information. This brings up issues concerning confidentiality, data leakage, and breach of compliance. Financial institutions should institute strict data-handling policies, encryption schemes, and access control measures so that the assessment of technical debt does not diminish any customer trust or regulatory compliance.

Resistance to Change and Culture.
Besides technical barriers, organizational culture is another factor hindering the automated assessment adoption. One would suggest that the development and operations teams, who have operated in a certain traditional way, simply would not welcome new tools and processes because such knowledge is perceived as disruptive and threatening. This would become quite difficult when the banking institution is strict in hierarchy and has a well-structured workflow, thus hindering the formation of continuous quality improvement culture. Change management, stakeholder involvement, training programs, and all other transaction activities would be the key to fighting resistance in the implementation-stage.

Cost and resources implications.

Automated technical debt assessment must be considered as an investment, so the required tools and infrastructure need to be purchased in addition to having the competent staff to execute it. Depending on their budget, some may see the price of enterprise-grade solutions, together with the expenses of integrating and maintaining them, as prohibitive. Furthermore, the banks should strive to employ skilled staff members able to create the tooling, interpret the results, and infer recommendations for an implementation plan. The decision-maker's reluctance to elevate technical debt evaluation beyond business requirements may arise in the mismatch between costs and benefits.

5. Case Examples / Applications

Banking in the real world.

Some financial institutions initially experimented with or implemented automated technical debt assessment methods to improve system reliability and modernization policies:
Large European Bank - CAST Highlight - Legacy Modernization a Large European bank did use CAST Highlight to assess several thousand applications in COBOL and Java. The tool was used to gain insights into software complexity and maintainability and cloud readiness so that the institution could decide on requiring change, platform change, or application retirement. This expedited the main banking modernization agenda and minimized operational risks.Continuous Monitoring with SonarQube at North American Bank: A U. S. financial institution employed SonarQube as part of its CI/CD pipeline to monitor and measure code quality and technical debt. Optimizing through assessment automation within development teams allowed the bank to keep code duplication minimum while maintaining extremely high security compliance and adherence to code standards among large-scale projects.Asian Bank-Predicting risks using machine learning: An Asian bank experimented with the machine learning method of technical debt assessment by training models over a sample of historical defect and incident data. This has engendered foresight as to where the technical debt is likely to arise the most, so the institution will be putting its investment into modules that are at highest risk to enhance system resilience and customer experience.These examples demonstrate how automated technical debt assessment is fast becoming a strategic enabler in digital transformation within banking.

Debt Prioritization Metrics and Indicators.

Automatic grading methods provide a plethora of metrics useful in decision-making, particularly in cases that require prioritization:

Complexity Metric: Cyclomatic complexity and m aintainability index can help to identify error-prone modules, which are indeed difficult to mainta in.

Duplication and Redundancy: Sometimes developers may write similar pieces of code - it is bad practice. There are also tools with duplicating code indicators which show where the developers prefer to copy and paste.

Lack Of Quality Control: This is the elimination of defects, taking into account the identification of defects in software.

Architecture Violations: Architectural debt is indicated by the presence of dependency cycles, violations of layering principles and excessive coupling.

Change Frequency (Hotspot Analysis): As modules are changed more often in the production environment, they being worked upon are riskier and they should be refactored.

Estimation of corrective action costs: SonarQube and CAST are software tools, by utilizing which a financial institution can define the resolution time of the identified issues and their maintenance costs, especially balancing them against the value that those issues have to the business.
The above parameters contribute to helping the banks to structure their debt repayments in a business centric, regulator compliant and congruent with the modernization agenda way.

6. Future Directions

AI and Predictive Analytics to handle Technical Debt.
In relation to the future of managing technical debt in current banking systems, it is expected that artificial intelligence (AI) and predictive analytics cannot play a less significant role. AI models can forecast areas that would tend to generate technical debt by employing previous failures data, system performance data and also changelogs. Predictive analytics can additionally be used very effectively in identifying high risk modules and therefore act on them prior to their being a serious impediment. Also, it is also possible to use text mining techniques, such as natural language processing (NLP) in order to find the gaps in documents

(which restrict the functionality of the systems), or even the application of reinforcement learning to find the optimal business-impact-oriented strategies for refactoring of the solutions.

Continuous check-out and live assessment.
Moreover, such monitoring would be indispensable if the overall accretion of technical debt is to be restrained. Continuous scanning would provide mechanisms to detect cognitive errors and other converging factors between the positions of architecture and code compliance. The automated check that provides ongoing monitoring of changes introduced into the development process and helps identify code compliance issues at the initial stages of software development and costs less than defect correction is referred to as shift-left testing. This is why throughout the migration of the legacy banking systems under renovation, continuous monitoring will help ensure that no unforeseen additional technical debt is acquired during the migration.

Compliance with Regulatory and Compliance Requirements.
The banking industry is facing challenges in the arena of regulatory issues. One of the latest ways in which technical debt is measured will be the new methodology converting legacy scanning with traditional software quality measures incorporated embedded in it. Some more standards are more the European General Data Protection Regulation (GDPR), the Payment Card Information Security Standard (PCI-DSS) and Basel III however many of solutions to resolve these requirements could be automated and might be refined to deal solely with these matters. In that case, reaching the Code Debt assessment as a remediation of compliance debts in software engineering and risk practices and standards violations will be used as a tool for exercising control over the quality and management of software and the risk administration.

Figure 1. Technical debt

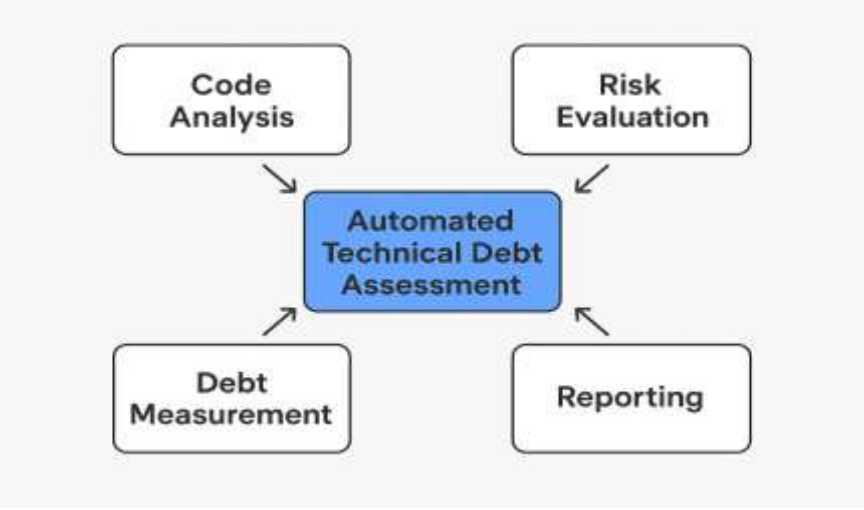

Figure 2. *A simple diagram showing Automated Technical Debt Assessment at the center, connected to four key processes: Code Analysis, Risk Evaluation, Debt Measurement, and Reporting*.

4. Conclusions

In the world of technology, one of the leading challenges faced by many financial institutions is old traditions. It becomes noticeable when analyzing the modernization of the banking environment during the past two decades. Old, outdated technologies, inflexible architectural approaches, as well as an archaic technological infrastructure also increase the costs of architecture as well as the support and sustain of the existing systems. These challenges can be successfully implemented with the help of automated technical debt assessment considering the fact that such techniques are relatively easy to implement, very accurate and can be used on a very large scale.

Traditionally, in the business of financial inclusion technical capabilities in banks have been weak with the market emphasizing developing products. Traditional financial institutions were and still can manage and service financial products such as savings, loans, and insurance without much integration with technology. 'Tech debt' is an industry recognized concept that cuts across the whole of IT and revenue governance of the business landscape. Steps must now be taken to resolve the above-described problems in technology used in financial institutions.

Finally, it is particularly important to manage technical debt with a view to coping effectively with changes that take place in the volatile banking environment. Financial institutions will eventually be able to strike that balance between the immediate need to cut costs and keep to the status quo and the huge challenges of the future through application of automation, forward looking techniques like predictive analytics and services which are provided all the time and are needed within a framework where security and flexibility are not compromised.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Biazotto, J. P. (2024). Technical debt management automation: State of the art. *Journal of Systems and Software*.

[2] Ciancarini, P., Falessi, D., Lenarduzzi, V., & Russo, B. (2020). The Strategic Technical Debt Management Model (STDMM). *Proceedings of the International Conference on Software Engineering*.

[3] Falessi, D., Izurieta, C., & Zazworka, N. (2020). An overview and comparison of technical debt measurement tools. *Empirical Software Engineering*, 25(5), 3830–3862.

[4] Gupta, R. K., Kumar, S., & Singh, P. (2016). A pragmatic approach for managing technical debt in legacy systems. *Proceedings of the ACM Symposium on Applied Computing*.

[5] Haki, K., Aier, S., & Winter, R. (2023). Digital nudging for technical debt management at Credit institutions. *Information Systems Journal*.

[6] Capco. (2023). How Capco automated legacy applications refactoring for a tier 1 bank. *Capco*. https://www.capco.com/about-us/success-stories/automated-legacy-applications-refactoring-for-a-tier-1-bank

[7] Khomyakov, I. (2019). Automated measurement of technical debt: A systematic literature review. *Journal of Software: Evolution and Process, 31*(11), e2195.

[8] Lenarduzzi, V., Taibi, D., & Janes, A. (2021). A systematic literature review on technical debt prioritization. *Information and Software Technology*, *128*, 106397.

[9] Monaghan, B. D. (2020). Redefining legacy: A technical debt perspective. In *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME)*. IEEE.

[10] Moreschini, S., Martini, A., & Bosch, J. (2023). Getting trapped in technical debt: A sociotechnical analysis. *MIS Quarterly*, *47*(3), 1431–1458.

[11] Nayebi, M., Lenarduzzi, V., & Falessi, D. (2024). Technical debt management: The road ahead. *arXiv preprint.*

[12] AlOmar, E. A., Christians, B., Busho, M., AlKhalid, A. H., Ouni, A., Newman, C., & Mkaouer, M. W. (2021). SATDBailiff: Mining and tracking self-admitted technical debt. *arXiv preprint arXiv:2107.00073*. https://arxiv.org/abs/2107.00073

[13] CodebTech. (n.d.). Managed services: A solution for successful digital transformation in legacy banks. *CodebTech*. https://www.codebtech.com/how-managed-

services-help-legacy-banks-streamline-digital-transformation/

[14] CodeScene. (2018). CodeScene: Behavioral code analysis tool that helps prioritize technical debt hotspots. *CodeScene Documentation*.

[15] Datasumi. (n.d.). GenAI: Optimizing legacy code migration in the banking industry. *Datasumi*. https://www.datasumi.com/genai-optimizing-legacy-code-migration-in-the-banking-industry

[16] Devox Software. (n.d.). Remove technical debt without slowing down. *Devox Software*. https://devoxsoftware.com/legacy-modernization/tech-debt-management-services/

[17] Forbes. (2022, May 24). Managing technical debt from legacy systems not moving to cloud. *Forbes*. https://www.forbes.com/sites/peterbendorsamuel/2022/05/24/managing-technical-debt-from-legacy-systems-not-moving-to-cloud

[18] Insight7. (n.d.). How to evaluate technical debt in contact center legacy systems. *Insight7*. https://insight7.io/how-to-evaluate-technical-debt-in-contact-center-legacy-systems/

[19] Li, Y., Soliman, M., Avgeriou, P., & van Ittersum, M. (2023). DebtViz: A tool for identifying, measuring, visualizing, and monitoring self-admitted technical debt. *arXiv preprint arXiv:2308.13128*. https://arxiv.org/abs/2308.13128

[20] Lumenalta. (2024). Reversing tech debt through legacy application modernization. *Lumenalta*. https://lumenalta.com/insights/legacy-application-modernization

[21] McKinsey & Company. (2020). Tame tech debt to modernize your business. *McKinsey Digital Insights*. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/breaking-technical-debts-vicious-cycle-to-modernize-your-business

[22] ScienceDirect. (2024). Technical debt management automation: State of the art and future perspectives. *Information and Software Technology, 161*, 107186. https://www.sciencedirect.com/science/article/pii/S0950584923002306

[23] SDV International. (n.d.). Tech debt: How to ease the burden of legacy systems. *SDV International*. https://www.sdvinternational.com/insights/tech-debt

[24] Sheikhaei, M. S., & Tian, Y. (2023). Automated self-admitted technical debt tracking at commit-level: A language-independent approach. *arXiv preprint arXiv:2304.07829*. https://arxiv.org/abs/2304.07829

[25] Shivashankar, K., & Martini, A. (2025). TD-Suite: All batteries included framework for technical debt classification. *arXiv preprint arXiv:2504.11085*. https://arxiv.org/abs/2504.11085

[26] Synchrony Systems. (n.d.). What is the true cost of technical debt in legacy applications? *Synchrony Systems*. https://sync-sys.com/what-is-the-true-cost-of-technical-debt-in-legacy-applications/

[27] Virtusa. (n.d.). Technical debt remediation: Improve your financial applications. *Virtusa*. https://www.virtusa.com/lp/technical-debt-remediation

[28] Webo.Ai. (n.d.). Technical debt management made easy with AI. *Webo.Ai Blog*. https://webo.ai/blog/technical-debt-management-made-easy-with-ai

[29] Wikipedia. (2019). Business rule mining. *Wikipedia*.

[30] Wikipedia. (2025). Technical debt. *Wikipedia*. https://en.wikipedia.org/wiki/Technical_debt