

Technical debt remediation challenges: The way forward



Table of **contents**

03 What is technical debt?

04 How technical debt gets accumulated

05 Post-technical debt effects

05 Technical debt remediation: key
success factors

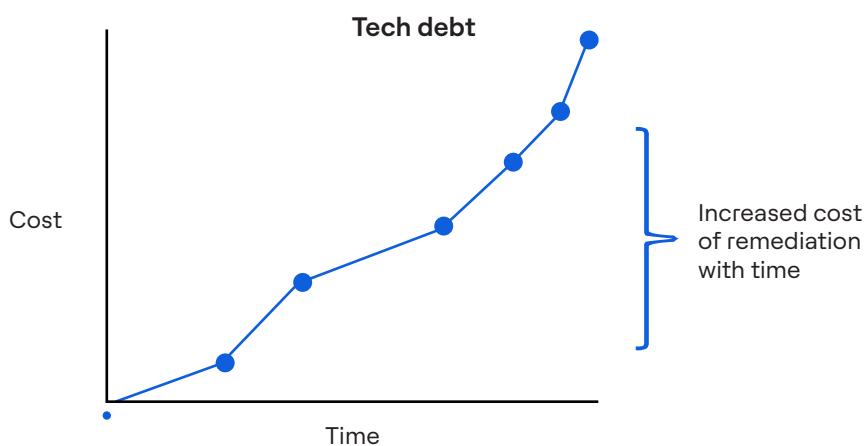
08 Conclusion

Change is the only constant in our ever-evolving world. The technology industry is driven by this rapid change, which is forcing financial services organizations to quickly adapt to stay relevant. Failure to adapt to the latest technology puts these organizations at risk of losing customers and becoming outdated. While many financial services organizations have already embarked on the digital transformation journey to keep up with advances in technology, not being fast enough may create a level of technical debt that can derail modernization efforts. This whitepaper discusses the effects of technical debt on the IT ecosystem and how financial services organizations can overcome them.

What is technical debt

During the development of an application, certain sections of code may get compromised due to various factors, such as technical limitations, framework constraints and the need for quick fixes. Such software or compromised codes must be refactored at a certain time to ensure continuous support for the application. This additional development work that arises from the requirement to upgrade software code and technology versions to prevent them from becoming obsolete is described as technical debt.

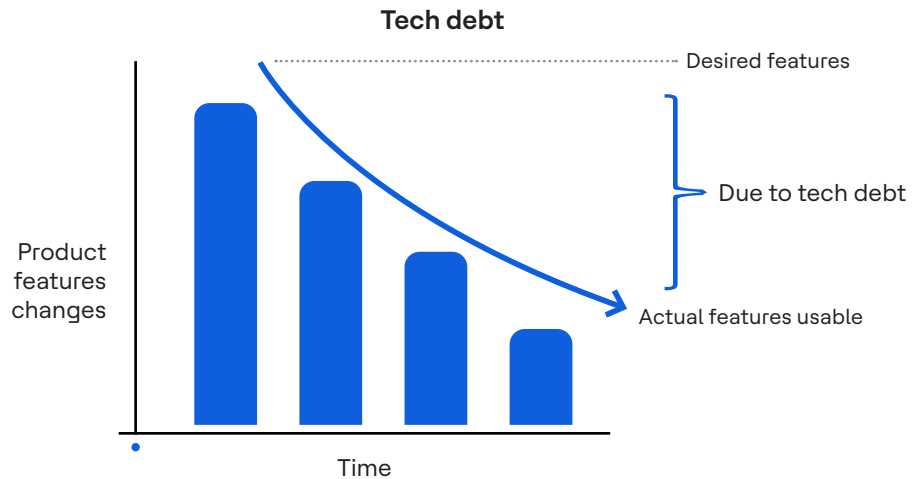
Similar to financial debt, which accumulates interest over a period of time, technical debt also increases the instability of the platform and makes it more expensive to maintain or refactor it.



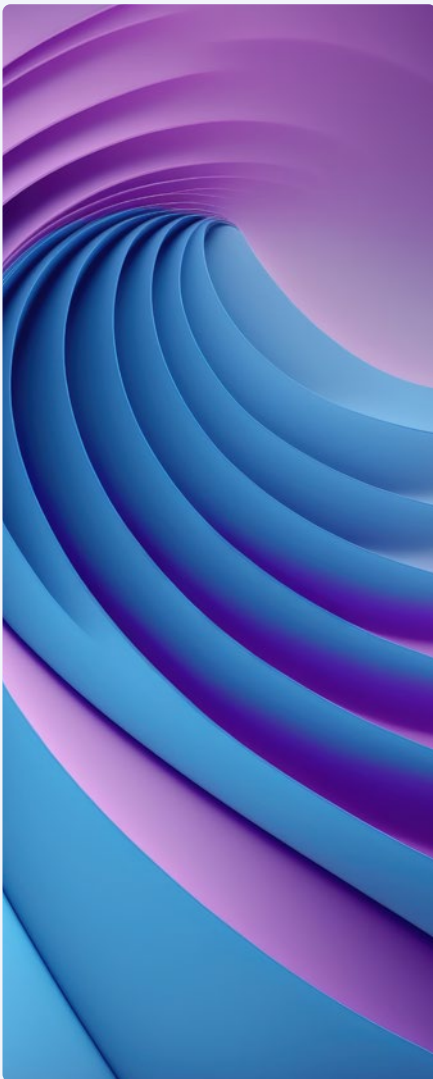
In general, technical debt refers to the accumulation of the rework that an organization has to address in the future in order to keep up with the current functional application performance or maintain the status quo.

Technical debt can occur either intentionally or unintentionally, and if left unaddressed, it can become increasingly challenging and costly to maintain the platform.

Adding more features and customizing the application are cumbersome tasks and pose a serious threat to user experience and customer satisfaction. Also, they could lead to operational risks and regulatory penalties. Applications in obsolete technology can cause issues to business continuity (BCP).



How technical debt gets accumulated



- Many banks and financial services institutions are still being powered by legacy technologies. This brings an additional angle to the tech debt problem, where the number of people with hands-on experience and knowledge of these legacy technologies is slowly declining.
- Many educational institutes and the new breed of engineers are embracing modern technologies like AI, cloud-based solutions and SPA (single-page apps). This is a serious concern as it will create an issue of skill availability to maintain the platform. Firms are in a fix as they are at the same time under tremendous pressure to modernize platforms to meet market demands.
- Legacy systems have been subjected to many changes (to address business needs, regulatory requirements, etc.) Over time, it has become increasingly challenging to maintain these systems or upgrade them. They carry huge technical debt, causing major technological bottlenecks in modernization and preventing the business from being competitive and innovative. In fact, most of the time is spent on addressing the complexity of the current problem rather than exploring new opportunities and cutting-edge technologies.
- Legacy systems were not designed with a holistic view from an architectural standpoint. They lack cohesiveness. These fragmented architectures create challenges for technology teams in upgrading the platform, thus limiting the ability of the business and its customers to leverage advancements in the technological landscape for improved decision-making and customer experiences.
- As banks merge with and acquire other banks, they often fail to accurately assess the extent of the technical integration that will be necessary, leading to further complexity associated with fragmented systems.
- The complexity increases when the legacy technology reaches its end of life, meaning that all systems updates and support once offered by the provider cease to exist. Deprecation of outdated solutions can cripple an enterprise and dramatically increase technical debt.
- When audited, financial services institutions are subject to huge penalties for using unsupported software.

Post-technical debt effects

Technical debt brings a host of issues that include:

- Increasing the cost of maintenance for platforms
- Disjointed architecture and limited scope for integration, thereby disabling a seamless STP in the customer and user journey
- Hurdles to adapting to digital transformation and scaling nimbly increase customer pain points
- Inability to meet changing customer expectations/needs/regulatory mandates
- Dearth of available skills
- Compatibility with cloud becomes a question and uninterrupted downtime becomes a distant dream

Technical debt remediation – key success factors

Customer satisfaction and an enhanced customer journey are the focus of most banks trying to stay competitive in the fintech space. While fintech firms already have a tech advantage, traditional banks are rushing to upgrade their tech stacks. Remediation has never been an easy transformation for banks considering the project timelines to which each and every program has been subjected. It is an uphill task that requires proper planning and a strategic approach to achieve successful outcomes.

Below are several recommended strategies and approaches that can help banks ensure the successful completion of their remediation journeys with limited hurdles.

Scope of functionalities

- The stepping stone for a successful remediation program is to have a clearly defined scope that is confirmed by all key stakeholders. Many functionalities/features may be redundant and thus no longer required in the remediated world.
- Identification of such functionalities will help to reduce efforts spent on validating the functional or technical features during the remediation journey:
 - Prioritization of functionalities/features that need to be remediated must be identified well ahead of the program.
 - Interconnected functionalities and interdependency between multiple features must be clearly identified and articulated as part of the requirements.
 - Because certain inbuilt features of the legacy platform won't be replicated in the new remediated journey, users must be informed that a 'like to like' match may not always be feasible, and an agreement must be made to this effect to ensure no critical functionality is compromised and nothing non-critical is included in the new technology.
 - These projects must be completed within stringent timelines. Hence, SLAs for clarifications, walkthroughs, sign-offs and board reviews should be adhered to strictly. Any slippages will have an impact on the overall timelines and project success.



Platform assessment/proof of concept

- Remediation programs will have to be addressed with a focused approach. Since fitment with the new technologies has its own risks, it is better to start with taking a small functionality and doing a proof of concept (POC).
- The POC should be the foundation of the framework. It will give a firsthand experience to refine the complexity of the platform and help to plan the future course of the program.

Analysis: design-led development

- Analysis of the legacy system and the corresponding interpretation should be presented in such a way that it can be easily visualized and understood by the design and development team. It will help them to factor the 'to-be' design appropriately without missing any existing functionalities. Any ambiguity or misinterpreted design can lead to the development of incomplete features.
- The design must be driven based on the following considerations:
 - Functionalities are going to be built 'as-is'
 - Whether screens are to be consolidated, business services are to be built, data layers are to be reused or any tools/converters are needed, etc.
- Dead code (no longer in use) in the legacy platform poses a serious challenge to the remediation work since there is no provision to test or simulate the functionalities (that are not used well). The design should have a proper workaround to handle/segregate any dead code.



Use of tools and accelerators for code conversion

- Tools and accelerators help reduce the manual efforts in any development, maintenance or even remediation programs. While we appreciate the effect these tools bring to the programs, we need to be mindful that the same tools that have been successful in one project might not deliver the same result in another project.
- Customizing the tools based on the nature of the program has to be done on priority. One needs to understand that tool customization will be an ongoing activity for the entire course of the program, and a tool that is being customized iteratively can help reduce the manual effort only to a limited degree.
- Need to ensure that proper time is available for refactoring code based on the output shared by the tools or accelerators.

Adequate skills

Although the focus during the remediation program is on newer technologies and skills, we need to ensure that there are plenty of available engineers with legacy skills. This will help expedite the legacy platform's analysis and reduce the turnaround times for requirements and technical analysis.

Necessary training programs must be planned well in advance to upskill or cross-skill available talent. Cross-skilling helps software engineers design and develop the architecture by leveraging new technologies, which will reduce the dependency on legacy skills in the future.



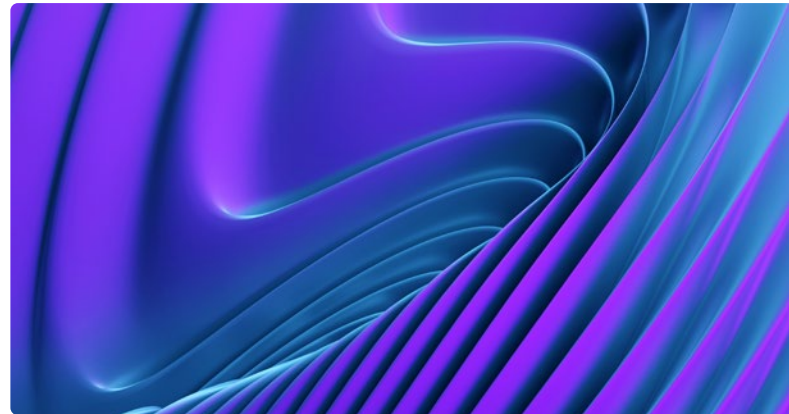
Infrastructure availability

- In any remediation program, the understanding of the existing system is the key factor for the transformation to be successful. It is always good to have a production replica of the lower environments so that analysis can be done appropriately. This will help reduce the risk of requirements being incorrect or unmet.
- If connectivity to the interfacing systems is not made available, the analysis will be incomplete and the end-to-end picture (flow of data/business rules) won't be able to be validated. Stubs can be used in the absence of connectivity to interface systems.
- Support from the platform ecosystem is also critical to ensure streamlined data flows, enhanced security and end-to-end visibility into the systems.

Scrum approach

Since most of the program implementations are moving towards an Agile-based approach, optimum care should be taken to ensure that the definition of done for each sprint is clearly defined based on the story points.

Remediation of the entire functionality/features of the epic in a single sprint might not be possible. Hence, it is important that the learnings of the previous sprints are considered while defining the velocity for future sprints.



Testing recommendations

- It is recommended to have an automated test suite to save manual efforts in comparing the functionalities of the legacy application with the remediated application.
- It is advisable to include test engineers during the analysis phase so that they can get an understanding of the legacy functionalities or features that will help them to come up with a better test strategy.

Other aspects to be considered



Scenario simulation

Some scenarios can't be simulated until the preceding steps/workflows are completed. Appropriate simulation techniques should be used so that the dependent features can be thoroughly analyzed and accordingly designed.



Traceability

Documentation for most of the legacy applications may not be up to date. So, tracing back the remediated features with legacy application-based features will be an uphill task. The appropriate methodology should be followed to mitigate the traceability risks.



Application co-existence

The legacy application and remediated application will co-exist until the legacy application is completely de-commissioned. So, there should be a planned mechanism to sync up the data between applications, and accessibility (for both applications) to end users should also be provisioned appropriately to avoid the downstream impact.

Conclusion

Technical debt is inevitable for banks and financial services organizations. While it may be unavoidable, if banks have appropriate strategies for managing technical debt effectively, research shows that they can free up as much as 50% of development time, effort and costs associated with managing the legacy application. Ultimately, they must periodically review their technical debt periodically and use an incremental approach to reduce the debt instead of remediating in one go.

Author bio

Vijaykrishnan Suchindrababu

He is a Practice Manager in the banking domain at HCLTech. He has 19+ years of experience in consulting, business analysis and product management. His domain area of expertise includes consumer banking, loans and mortgages. He has worked with banks across geographies and has been part of several core banking and mortgage product implementations and transformation journeys.

Gomathi Madheswaran

She is a Technical Architect at HCLTech. She has 15+ years of experience in Oracle consulting with technical and functional expertise across the banking, telecom, IT and retail industries.

HCLTech
Supercharging Progress™

www.hcltech.com

Copyright © 2024 HCL Technologies Limited