LTI
Let's Solve

A Larsen & Toubro Group Company

Point of View

Technical Debt: The Slippery Devil

by **Mahesh Bondre**

Large organizations often need to build some applications in-house, buy commercial-of-the –shelf (COTS) products, acquire companies and/or own new applications. Vendors build applications for IT departments, but as time progresses, the vendor could change, and the same vendor may not be the one supporting / enhancing/ fixing the defects. Such applications tend to become a liability than assets due to the accumulated technical debt. Like monetary debt, technical debt can accumulate "interest" and result in a loss of money for the organization.

Several organizations are facing the financial impact – in terms of cost of quality and delay – of technical debt that has accumulated leading to slow systems, delay in releasing new features and increase in defects in the system. To counter this, many organizations have slowed down product releases, and are applying the 6Rs (Re-host, Re-Platform, Replace, Retain, Retire, and Refactor) to their old systems. For critical applications, the most useful lane is Refactor, as it will not only reduce the technical debt, but also improve the performance of the systems.

In today's VUCA (Volatility, Uncertainty, Complexity and Ambiguity) environment, faster time-to-market and quality of deliverables and customer experience, are critical. All products or services need to hook users with these three factors, else there is a risk of losing them to competition. To achieve competitive excellence, organizations have started using Agile (where adding the code every sprint on top of the previous code) and DevOps methodologies. However, technical debt continues to be the slippery devil. Here are some reasons why this happens.

Code smells

"Code smells" is an age-old term that refers to code that is old and needs to be refactored to improve the performance. Even if you visit your code written couple of week / months back, you will get the feel that the code is inefficient.

We're too busy

If you ask any developer, what is the cause of increase in technical debt the immediate response is "committed tight deadlines for the release".

Apart from the poor planning, bad coding practices, limited test coverage, inexperience with a specific technology, legacy technology and monolithic architectures, and long pending modernization are the major contributors to high technical debt.

Common symptoms of technical debt

1. Increased defect count
2. Degraded performance of the system
3. Higher consumption of resources like CPU / memory
4. Complexity of the systems

Symptoms indicate an infection, but tests underline the depth of the infection. These symptoms will tell you that there is technical debt, but AI ops tools like Code Analysis Tools by CAST will conduct a detailed analysis of the code and measure the depth of technical debt.

It is everyone's responsibility to address technical debt to avoid a snowball effect, which will require the team to spend extra time in fixing issues, thus taking their focus away from new development work. Longstanding technical debt also lowers the performance of the system and increases the maintenance cost.

Technical intervention

Techniques like peer reviews or pair programming help reduce technical debt, but today, developers can choose from several tools for continuous risk monitoring to detect structural defects in code early on, fast feedback to prevent defects, continuous visibility into system performance, safety, and reliability. Some tools test every line of code and provide the detailed root cause of each defect, making it easy to fix it quickly.

These tools enable developers to write a clean code by following coding standards, check many aspects of code and identify class design problems, method design problems etc.

Couple of tools also give the estimates in hours/days to fix this technical debt. These tools have made the developer's life easy. Some of the more popular tools are:

- Coverity
- SonarQube
- Checkstyle
- Closure Compiler

The best solution is enforcing the coding disciplines using the ADKAR Model:

ADKAR	Legacy System	New Technologies
Awareness	Run the tools like CAST to generate technical debt report	Train the team on coding standards
Desire	Take a pause on the release and initiate the modernization initiatives – 6R Lanes	Following the coding standards and using tools to make sure the same are enforced
Knowledge	Train the teams on various tools and precautionary measures to avoid it to further	Keep up to date with the new tools and techniques for faster deliveries
Ability	Convert monolithic to microservices architecture	Hardening/ stabilization sprint to complete the activities
Reinforcement	Apply new ways of working, regular analysis and removal of technical debt	Effective use of continuous integrations and AI/ ML tools

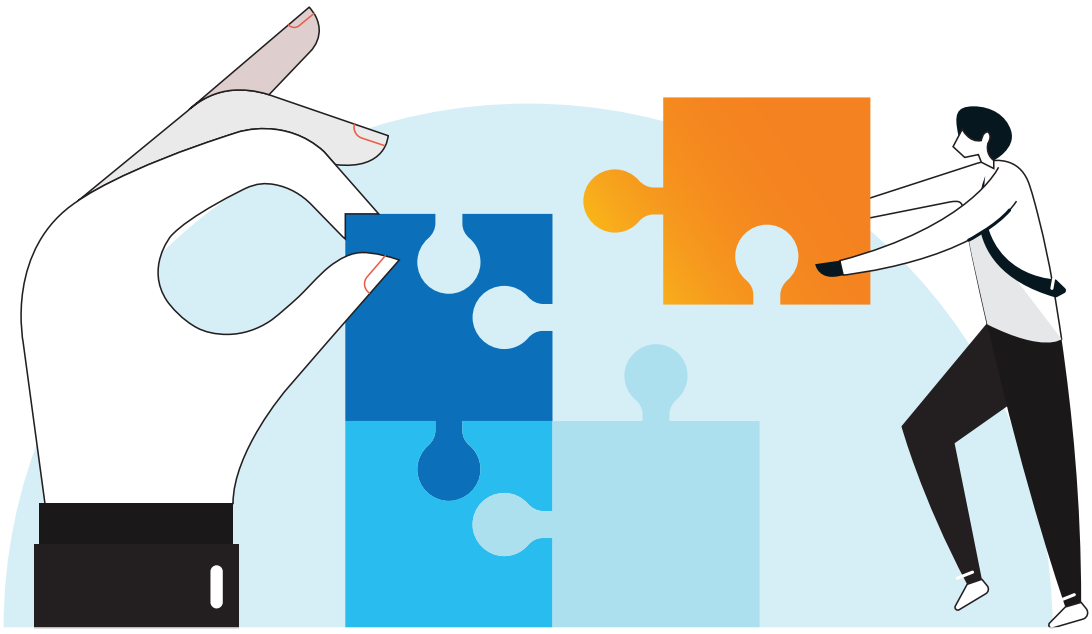
Source: https://www.prosci.com/adkar/adkar-model

Final thoughts

Technical debt is a slippery devil, which often goes neglected in organizations, however the snowball effect that it creates can bring the biggest of organization on knees. A highly profitable business can run into losses with sheer power of technical debt. It is imperative for organization to treat early signs of technical debt and start taking corrective actions as it have significant effects on cost of quality and cost of delay. The technical debt can be corrected/avoided if teams continue applying coding standards/ disciplines, effective use of tools and sustain the change by applying models like ADKAR.

As put simply by Michael Feathers(author. of the book Working Effectively with Legacy Code): "Clean code looks like it was written by someone who cares." There's a bunch of low-level technical details, but the goal of clean code is to encourage programmers to care, and to have pride in their workmanship.

Source: Clean Code for Managers: Robert C. Martin (https://codingsans.com/blog/clean-code)



LTI
Let's Solve

Author Profile

Mahesh Bondre

Associate Principal, Consulting , LTI

Mahesh has 20+ years of experience in IT delivery management, last 10 years in Agile and DevOps transformation. He has played various roles like Delivery Manager, Scrum Master, Agile Coach and Agile CoE Head. He has an excellent record of working with clients in Insurance and Banking domains across the globe.

Under his leadership, the team has developed various (IPs) assessment tools, Next Gen Agile frameworks and filed a patent too. These tools, frameworks and case study are recognized by Gartner and published in their report worldwide.

info@Lntinfotech.com