happiest minds
The Mindful IT Company
Born Digital . Born Agile

MANAGING TESTING DEBTS IN SOFTWARE PROJECTS

WHAT IS TECHNICAL DEBT

In software development, technical debt is the implied cost of additional rework caused by choosing an easy (limited) solution instead of using a better approach that would take longer to implement. Analogous to monetary debt, if technical debt is not addressed properly, it can accumulate "interest", making it harder to implement changes. Unaddressed technical debt increases the overall cost of software development. In other words, it's the result of prioritizing speedy delivery over perfect code. Technical debt results in teams who make sacrifices in design and jump into coding. Technical debt can occur in any corner of IT. The more the debt or ignored issues build, the costlier it is to rectify.

A majority (69%) of IT leaders say technical debt poses a fundamental limit on their ability to innovate, along with 61% felt it drags on their company's performance & it will continue to have a major impact in the future.

Technical debt may have one or more causes such as:

- Time pressures
- Overly complex technical design
- Poor coding standards
- Lack of skill
- Suboptimal code
- Delayed refactoring
- Insufficient testing

Like Technical debt, we have Testing debt which is quite prevalent across Project teams in Software Development Projects
Testing debt occurs when a team intentionally or unintentionally chooses an option that yields benefit in the short term, but results in accrued testing costs in terms of time, effort, or risk in the longer term.

In simple words: testing debt is all about the testing that you should have done but for some reason, you didn't. Some of the prime causes for Testing debt are:

- Untested features
- Tests that are slow or impractical, which can lead to being forgotten/not performed
- Items/components that are complex to test
- Tests you cannot rely on

Just like accumulating laundry, manually testing every major or minor release can quickly get out of control. With every release, your technical debt grows, and so does the risk of something going wrong.

TEST DEBT UNDER MANUAL TESTING

Test engineers execute test cases manually by following the steps described in a test specification document. Here are the list of common contributors to test debt related to manual testing.

Limited test execution

Due to crunched Project timelines, the resources and timelines required to carry out comprehensive testing are inadequate. This results in the testing team executing only a subset of tests for a release, which might result in the possibility of defects getting leaked to production, thereby compromising on the overall quality.

Improper test design

Manual testing is a time-consuming and tedious process. A test case could be designed with many variations using different combinations of data. Since the execution of all combinations of test cases is a laborious process for testers, they restrict themselves to a few happy or positive scenarios during test execution. This increases the risk of residual defects in the System Under Test (SUT).

Missing test reviews

Test reviews help improve test cases' quality and help find the problems earlier in the life cycle. Missing test reviews could result in uncovered test cases, or edge case scenarios which might result in defect leakage to production, and the overall quality of the product might get compromised.

TEST DEBT IN AUTOMATION TESTING

Automation testing involves executing pre-scripted tests to execute and check the results without manual intervention. The list of factors that contribute to test debt in automation testing is as follows

Inappropriate or inadequate infrastructure

Tests conducted in Environment/ infrastructure not resembling the customer's infrastructure could lead to unpredictable software behavior, resulting in reduced confidence in the system.

Lack of coding standards

Automated tests must adhere to common coding standards. When the best practices with respect to coding are not adopted, it increases the effort taken to maintain the automation scripts and the scripts tend to fail, which increases the overall **Test Automation debt** over multiple release cycles.

Unfixed (broken) tests

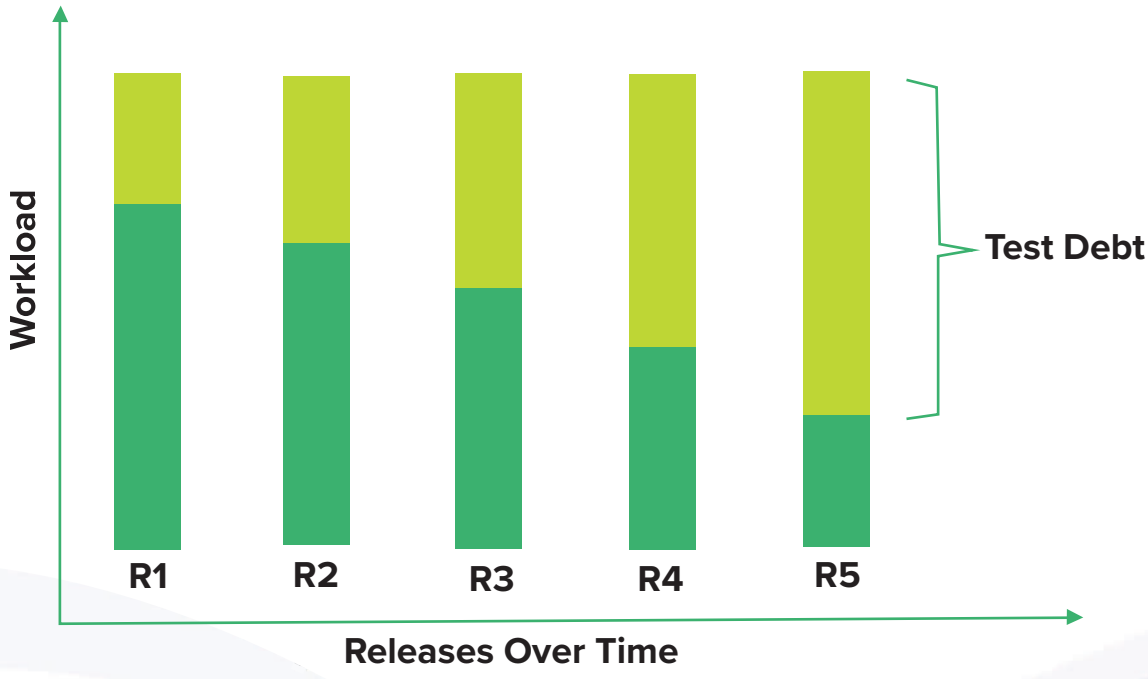
Not fixing broken tests or tests not being updated when functionality changes, reduces user confidence in the scripts, decreases the quality, and increases the overall maintenance effort, which needs to be factored in over subsequent releases to avoid automation debt.

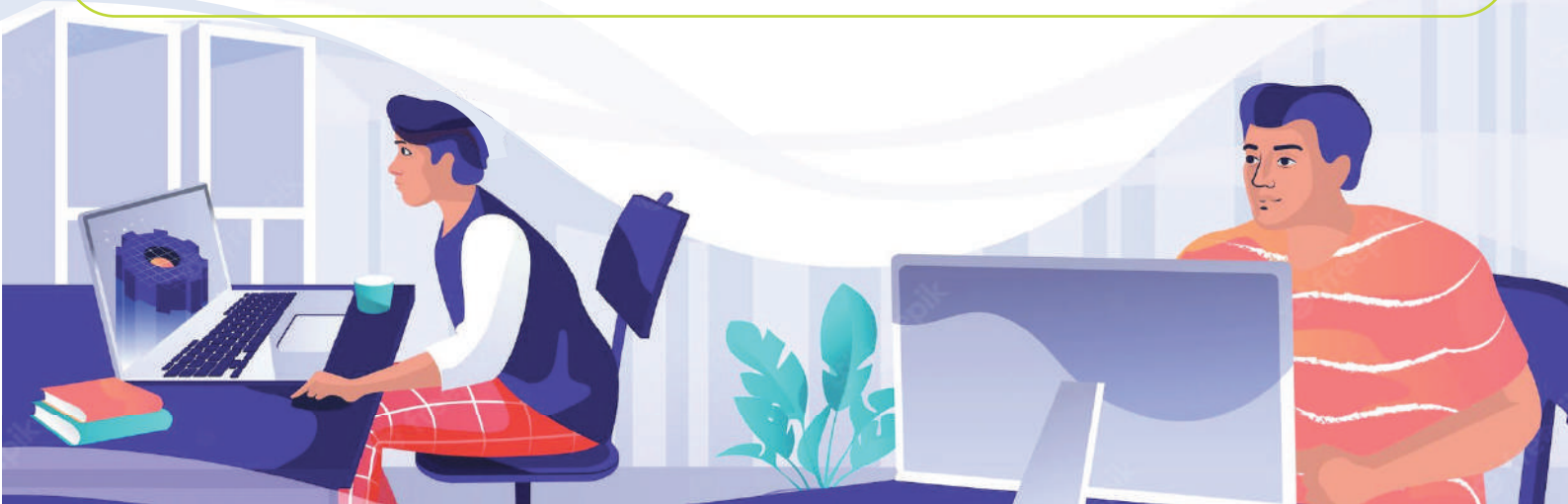
Using record and replay

It is easy to use record and replay tools for testing Graphical User Interface (GUI) applications. Record and playback tools and plugins are never a long-running solution since these have minimal test coverage and complex workflows cannot be validated using record & Playback tools. This approach reduces the test automation coverage, and the **backlog continues to pile up.**

Lack of Maintenance

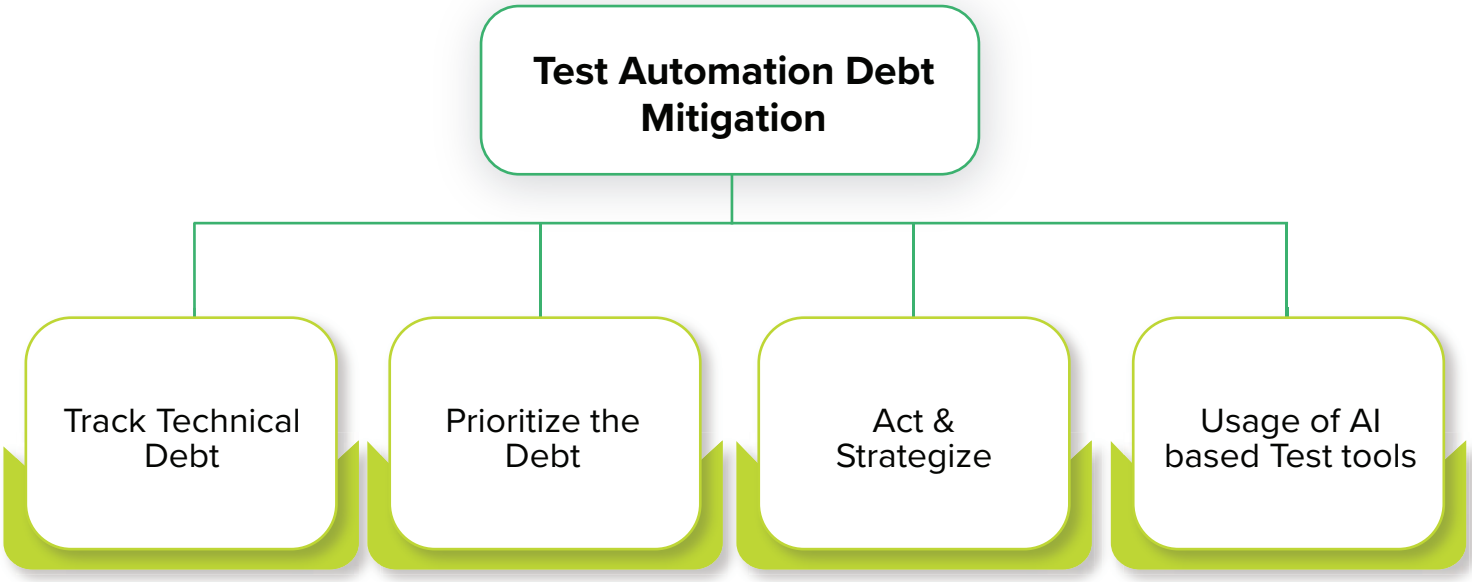
Test scripts rely on selectors (like element ID or Xpath) to find elements in your UI. For instance, you can instruct the script to click on a specific button or enter text in a field. The challenge is that these selectors change unpredictably as the application changes. Even simple styling or layout changes can result in the script choosing the incorrect button or entering text in the wrong field. This will result in flaky tests & if these are not updated frequently, the automation scripts will get into cold storage, and the **Automation debt** will keep accumulating over time.





HOW DO WE MITIGATE TEST DEBT?

Once we identify technical debt in our test automation process, it is essential to track, communicate and plan accordingly.



Track

Work with your management and plan to allocate a portion of the future effort to reduce the technical debt; it's imperative to identify and add some test debt stories during the sprint.

Prioritize

Prioritize work to reduce technical debt in test automation which will have the most significant impact. Have a strategy to automate the acceptance and sanity tests early and then have the regression suites automated.

Act

Plan the entire automation strategy & accordingly prioritize your most significant technical debts and invest time in fine-tuning the Framework design and coding standards. Sloppy test automation will increase the technical debt over a period.

Advanced Test Automation Tools & Frameworks

The project teams can leverage AI-based test automation tools to speed up the entire test script design. This will increase the test coverage and reduces the time created to create tests. These AI-based tools also have self-healing capabilities which fix the tests whenever there is a change in locators in the application. This will vastly improve the maintenance effort required to fix the automation scripts. This will help the automation testers to cover the ground rapidly which will help teams to reduce the overall debt in test automation.
It is an undeniable truth that the more you invest in design and coding standards for test automation, the more significant the long-term payback you will receive. Shortcuts and sloppy test automation will have commercial implications over a period.

FINAL THOUGHTS

Many test automation teams struggle with the need for speed in agile development. Some teams use unstructured record and playback methods or inaccurate testing tools to create and automate tests quickly, resulting in “throw-away automation”.

“Just because your test automation is working doesn’t mean that it adds value.”

CONCLUSION:

In conclusion, testing debt is a common issue in software projects, and can have a significant impact on the maintainability and overall health of a codebase. Addressing testing debt requires a regular testing process, a strong focus on test automation, and maintaining test infrastructure. By addressing testing debt early, it can help to ensure that a project remains healthy and maintainable over the long term.

It's important to ensure that tests are independent from the implementation and can be run in isolation. This will make the tests more maintainable over time, and will make it easier to understand the impact of changes on the codebase.

Usage of Test metrics: Use testing metrics such as code coverage, test pass rate, and test execution time to track the health of the test suite. This can help to identify areas of the codebase that need more attention and can also help to identify tests that are no longer needed.

By implementing these best practices, it can help to minimize the impact of testing debt on the overall health and maintainability of a project, and ensure that tests continue to provide accurate and reliable results over time

AUTHOR BIO

Prashanth TV is a Practice Director at Happiest Minds with over 17 years of experience in the area of Software Testing & Quality Assurance and leads the QA practice within DBS unit of Happiest Minds. He has extensive experience in areas of Test Automation across multiple tools and technologies. He has worked on multiple domains including Airlines, Retail and Manufacturing. He also possesses wide experience in the consulting and pre-sales areas with a keen interest in emerging Digital Technologies.

About Happiest Minds

Happiest Minds Technologies Limited (NSE: HAPPSTMNDS), a Mindful IT Company, enables **digital transformation** for enterprises and technology providers by delivering seamless customer experiences, business efficiency and actionable insights. We do this by leveraging a spectrum of disruptive technologies such as: **artificial intelligence, blockchain, cloud, digital process automation, internet of things,** robotics/drones, **security, virtual/ augmented reality,** etc. Positioned as 'Born Digital . Born Agile', our capabilities span digital solutions, infrastructure, product engineering and security. We deliver these services across industry sectors such as automotive, BFSI, consumer packaged goods, e-commerce, edutech, engineering R&D, hi-tech, manufacturing, retail and travel/ transportation/ hospitality.

A Great Place to Work-CertifiedTM company, Happiest Minds is headquartered in Bangalore, India with operations in the U.S., UK, Canada, Australia and Middle East.

Business@happiestminds.com

happiest minds
The Mindful IT Company
Born Digital . Born Agile

www.happiestminds.com