To:	Functionize, Inc(gary@functionize.com)
Subject:	U.S. Trademark Application Serial No. 97120746 - TEST DEBT
Sent:	August 23, 2022 06:55:30 PM EDT
Sent As:	tmng.notices@uspto.gov

Attachments

screencapture-testuff-com-software-testing-debt-crisis-16612937428971
screencapture-qeunit-com-blog-test-debt-fundamentals-what-why-warning-signs-16612937786981
screencapture-www-kiwiqa-com-understanding-technical-debt-in-software-testing-16612945633781
screencapture-www-functionize-com-blog-test-debt-explained-why-your-team-has-no-spare-test-capacity-16612952030201

United States Patent and Trademark Office (USPTO)
Office Action (Official Letter) About Applicant's Trademark Application

U.S. Application Serial No. 97120746

Mark: TEST DEBT

Correspondence Address:
FUNCTIONIZE, INC
1255 TREAT BLVD SUITE 300
WALNUT CREEK CA 94597 UNITED STATES

Applicant: Functionize, Inc

Reference/Docket No. N/A

Correspondence Email Address: gary@functionize.com

NONFINAL OFFICE ACTION

The USPTO must receive applicant's response to this letter within six months of the issue date below or the application will be abandoned. Respond using the Trademark Electronic Application System (TEAS). A link to the appropriate TEAS response form appears at the end of this Office action.

Issue date: August 23, 2022

The referenced application has been reviewed by the assigned trademark examining attorney. Applicant must respond timely and completely to the issues below. 15 U.S.C. §1062(b); 37 C.F.R. §§2.62(a), 2.65(a); TMEP §§711, 718.03.

SEARCH OF USPTO DATABASE OF MARKS

The trademark examining attorney searched the USPTO database of registered and pending marks and found no conflicting marks that would bar registration under Trademark Act Section 2(d). 15 U.S.C. §1052(d); TMEP §704.02.

However, applicant must respond to the issues set forth below.

SUMMARY OF ISSUES:

- Section 2(e)(1) Refusal - Merely Descriptive
- Identification of Services - Partially Indefinite

SECTION 2(e)(1) REFUSAL - MERELY DESCRIPTIVE

Registration is refused because the applied-for mark merely describes a purpose of applicant's services. Trademark Act Section 2(e)(1), 15 U.S.C. §1052(e)(1); *see* TMEP §§1209.01(b), 1209.03 *et seq.*

A mark is merely descriptive if it describes an ingredient, quality, characteristic, function, feature, purpose, or use of an applicant's goods and/or services. TMEP §1209.01(b); *see, e.g.*, *In re TriVita, Inc.*, 783 F.3d 872, 874, 114 USPQ2d 1574, 1575 (Fed. Cir. 2015) (quoting *In re Oppedahl & Larson LLP*, 373 F.3d 1171, 1173, 71 USPQ2d 1370, 1371 (Fed. Cir. 2004)); *In re Steelbuilding.com*, 415 F.3d 1293, 1297, 75 USPQ2d 1420, 1421 (Fed. Cir. 2005) (citing *Estate of P.D. Beckwith, Inc. v. Comm'r of Patents*, 252 U.S. 538, 543 (1920)).

In the present case, applicant has applied to register the mark "TEST DEBT" for use in connection with "AI-powered software as a service (SAAS) testing platform."

According to the attached evidence, the wording "test debt" (or technical debt) commonly refers to, generally, the needs of software test automation surpassing the time or capacity available to meet those needs." *See attached evidence.* Applicant's website states, "Functionize has been an AI company from day one. Our approach has been to apply machine learning and other AI approaches to solve the problems with *test debt*." *(Emphasis added) See attached evidence.*

Therefore, the mark TEST DEBT, merely describes the purpose of applicant's services, namely, to improve test automation by reducing testing maintenance and increasing overall test coverage. Accordingly, the proposed mark is merely descriptive, and registration is refused on the Principal Register under Section 2(e)(1) of the Lanham Act.

Although applicant's mark has been refused registration, applicant may respond to the refusal by submitting evidence and arguments in support of registration.

SUPPLEMENTAL REGISTER - ADVISORY

The applied-for mark has been refused registration on the Principal Register. Applicant may respond to the refusal by submitting evidence and arguments in support of registration and/or by amending the application to seek registration on the Supplemental Register. *See* 15 U.S.C. §1091; 37 C.F.R. §§2.47, 2.75(a); TMEP §§801.02(b), 816. Amending to the Supplemental Register does not preclude applicant from submitting evidence and arguments against the refusal(s). TMEP §816.04.

IDENTIFICATION OF SERVICES

The identification for software in International Class 42 is indefinite and must be clarified to specify the purpose or function of the software and its content or field of use, if content- or field- specific. *See* 37 C.F.R. §2.32(a)(6); TMEP §§1402.03(d), 1402.11(a). The USPTO requires such specificity in order for a trademark examining attorney to examine the application properly and make appropriate decisions concerning possible conflicts between the applicant's mark and other marks. *See In re N.A.D. Inc.*, 57 USPQ2d 1872, 1874 (TTAB 2000); TMEP §1402.03(d).

Applicant may adopt the following identification, if accurate (please note that suggested amendments are in bold italics and suggested deletions are in strikethrough):

International Class 42

"AI-powered software as a service (SAAS) testing platform ***featuring software for __****{indicate function of software, e.g., database management, etc.}"*

Applicant's services may be clarified or limited, but may not be expanded beyond those originally itemized in the application or as acceptably amended. *See* 37 C.F.R. §2.71(a); TMEP §1402.06. Applicant may clarify or limit the identification by inserting qualifying language or deleting items to result in a more specific identification; however, applicant may not substitute different goods and/or services or add goods and/or services not found or encompassed by those in the original application or as acceptably amended. *See* TMEP §1402.06(a)-(b). The scope of the goods and/or services sets the outer limit for any changes to the identification and is generally determined by the ordinary meaning of the wording in the identification. TMEP §§1402.06(b), 1402.07(a)-(b). Any acceptable changes to the goods and/or services will further limit scope, and once goods and/or services are deleted, they are not permitted to be reinserted. TMEP §1402.07(e).

For assistance with identifying and classifying goods and services in trademark applications, please see the USPTO's online searchable *U.S. Acceptable Identification of Goods and Services Manual*. *See* TMEP §1402.04.

RESPONSE GUIDELINES

How to respond. Click to file a response to this nonfinal Office action.

ASSISTANCE

Because of the legal technicalities and strict deadlines of the trademark application process, applicant is encouraged to hire a private attorney who specializes in trademark matters to assist in this process. The assigned trademark examining attorney can provide only limited assistance explaining the content of an

Office action and the application process. USPTO staff cannot provide legal advice or statements about an applicant's legal rights. TMEP §§705.02, 709.06. See Hiring a U.S.-licensed trademark attorney for more information.

Please call or email the assigned trademark examining attorney with questions about this Office action. Although an examining attorney cannot provide legal advice, the examining attorney can provide additional explanation about the refusal(s) and/or requirement(s) in this Office action. *See* TMEP §§705.02, 709.06. The USPTO does not accept emails as responses to Office actions; however, emails can be used for informal communications and are included in the application record. *See* 37 C.F.R. §§2.62(c), 2.191; TMEP §§304.01-.02, 709.04-.05.

/Kimberly Parks/
Kimberly Parks
(571) 272-6129
kimberly.parks@uspto.gov

RESPONSE GUIDANCE

- **Missing the response deadline to this letter will cause the application to abandon.** The response must be received by the USPTO before midnight **Eastern Time** of the last day of the response period. TEAS maintenance or unforeseen circumstances could affect an applicant's ability to timely respond.

- **Responses signed by an unauthorized party** are not accepted and can **cause the application to abandon**. If applicant does not have an attorney, the response must be signed by the individual applicant, all joint applicants, or someone with legal authority to bind a juristic applicant. If applicant has an attorney, the response must be signed by the attorney.

- If needed, **find contact information for the supervisor** of the office or unit listed in the signature block.

Software Testing Debt Crisis

I'm sure you are familiar with the term ***Technical debt***. We're all familiar with the ever going debates when designing and programming software – should we do it the 'right (and long, expensive) way' or maybe we should take the easier, faster road, which will get us quickly to a working new feature/product but also to a code that will need fixing or change in the future (therefore the 'debt').

It's the same when we look at **Software testing**.
Our testing projects present similar challenges as developers face:

- No Holistic, methodology-backed, approach to testing
- Not enough time to test
- No documentation
- Management not invested in testing, unaware of importance
- Not enough resources (mainly human – testers)
- Lack of expertise
- Testing group not involved in the design and requirements phase, therefore missing the big picture
- Testing pushed back and testing time is always the first to be shortened when business pressure starts

Therefore, they are also faced with what can be referred to as ***Software testing debt***.

Test Debt Types

In this process, we know that whatever we don't do now, do only partially, or do via the shortcut rather than the right path, all these will haunt us in the future. There's no right or wrong, it is all a matter of priorities and compromises. The important thing is awareness – making decisions while knowing the facts, not ignoring them, considering future consequences and aligning with the organization's priorities and business decisions.
Here are a few possible areas to consider as possible future ***Software testing debt***:

1. Full coverage of tests

1. Full coverage of tests
2. Integrating unit testing as an integral and obligatory part of the development
3. Implementation of functional testing as an integral and binding part of the release of the version
4. Implementation of automated testing as an integral part of the CI-CD process
5. Complete working methodology of development-testing-release
6. Measuring the work process of development-testing-bug fixing

Managing software Testing Debt

The first thing to understand is that the optimal point of the organization is ***not*** zero testing debt. This is crucial to remember when managing your *testing debt* (same goes for the concept of *Technical debt*). It's OK, and even recommended, to knowingly take measured risks, in order to gain advantage elsewhere (for example time saving).
When allocating time for dealing with your *Software debt*, try to find those elements that will have a positive impact on your project/product. Avoid those that won't make a real difference for the software's behavior or quality, even if it's something that bothers you – as a tester.
Also remember that you don't have to fix or sort out such a debt completely. Fixing it partially, so that the problem is less critical now, may be enough in many cases. Realizing this, can help you deal with a few issues, rather than allocating all the time to one of them only.
Working to minimize the created debt can be done in different ways:

- Dedicated team, working to solve and fix debt issues
- Allowing time for *debt fixing* at the end of each cycle/feature/project, done by the same team of testers
- Opportunity based fixing: whenever there's time, whenever there's a tester with free time, whenever there's a chance to fix something due to a new cycle/project, etc.

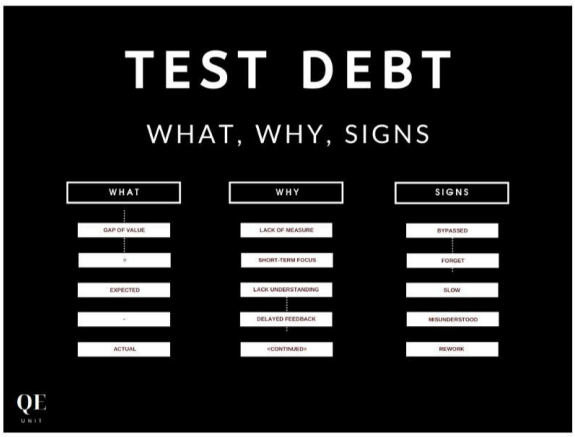
Each of these approaches has its advantages and disadvantages, choose whichever is best **for your specific needs and environment**.

There's one other thing that can help, and that is transparency. Yep, being transparent with other groups in the project, sharing your list of *testing debt issues* can be helpful in the future, since the codes, the products, are constantly changing. So are processes, priorities and even resources. Some of the issues, if known to everyone, may be dealt together with these ongoing changes, as part of the workflow, rather than as special treatment.
And lastly, differentiate between your manual and your automated testing with regards to software testing debts management. The factors are different and so should be their management.

January 22nd, 2019 | Blog, Development, Testing | 0 Comments

0 COMMENTS

Test Debt Fundamentals: What, Why & Warning Signs

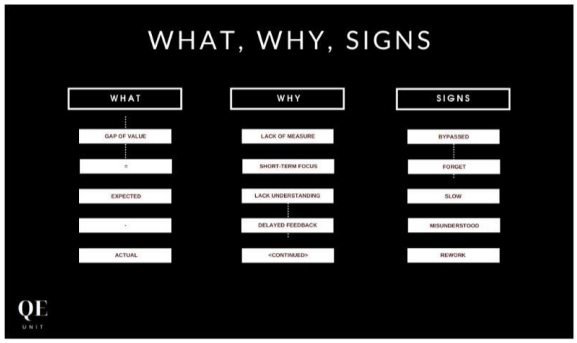


Antoine Craske August 3, 2021 Quality Engineering Perspectives 0

The decision is taken, “we are going to address our test debt”. It sounds positive, but at the same time we wonder, “why did we end up at that point?”.

Antoine Craske August 3, 2021 Quality Engineering Perspectives 0

The decision is taken, “we are going to address our test debt”. It sounds positive, but at the same time we wonder, “why did we end up at that point?”.



Antoine Craske August 3, 2021 Quality Engineering Perspectives 0

The decision is taken, “we are going to address our test debt”. It sounds positive, but at the same time we wonder, “why did we end up at that point?”.

Test Debt is a waste that organizations must continuously prevent and eliminate to maintain their speed and reactivity. Else, each change comes with its added weight, slowing down the overall software delivery.

During the round-table “Forget Test Debt: Build Test Automation Wealth” we shared the definition and key reasons for test debt before sharing actionable ways to build test automation wealth in the first place. This first article focuses on Test Debt, while the next one on creating test automation wealth.

I thanks each of the participants for their participation and contribution:

- **Joel Oliveira**, Software Testing & Quality Assurance Evangelist, Public Speaker, Trainer, Senior Program Manager, Independent Consultant
- **Rafael Amaral**, Senior Software Engineer in Test at Farfetch
- **Filipa Nogueira**, Engineering Team Lead at La Redoute

First things first, let’s start by exploring Test Debt with a concrete example

First things first, let's start by exploring Test Debt with a concrete example.

What is Test Debt anyway?

A company judges too slow its software changes iteration. Test automation is identified as a priority to accelerate the cycles. A framework is quickly built to automate the existing manual test and produce early wins. Then problems start to appear.

The precipitation led to an automated test suite not working well: many tests are executed with slowness and instability, while it is hard to implement some manual tests. In addition, the only person knowing the framework will leave the company. You open the tests to try to help but they are too technical and hard to understand.

You hoped for much better results. A fast and reliable automated test suite on the most valuable tests for the team. A framework known, maintainable by other persons or a vendor. A built-in reporting on shared team objectives and measures.

Technical Debt is about: Expected Asset Value – Actual System Value

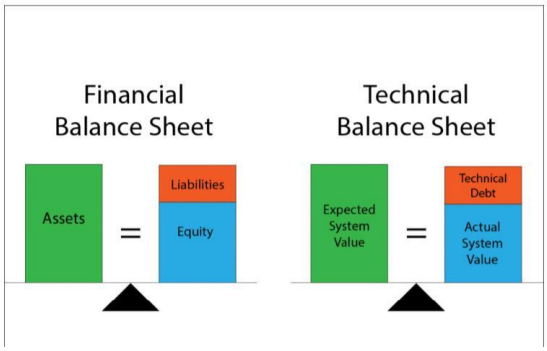


Figure 1: The analogy between Financial and Technical Debt https://pvpros.com/technical-debt/

Technical Debt and Test Debt share the common characteristics of being a gap between an expected and actual value. The Test Debt can be due to the lack of specific actions or be the result of particular decisions. It is important to keep these two typologies in mind as counter-intuitively, both inaction and action lead to debt.

The presence of Test Debt leads to the next question, why did we get there?

Why do we end up with Test Debt?

We can start by looking out of the world of software and testing, especially in finance, reusing our analogy. Why do so many people endup in a financial debt situation? Common reasons are shared with Test Debt.

A first point circles back to the two elements of measuring test debt, "expected" and "actual". The expected value means that it was defined, shared, and agreed upon. Unfortunately, the quality expectations are not yet a reflex of collaborative exercises, such as the Shift-Up one. The "actual" value lies in a characteristic of Test Debt: it is hard to measure.

> "The tricky thing about technical debt, of course, is that unlike money it's impossible to measure effectively."
>
> Martin Fowler, at martinfowler.com

Financial debt is easily measured; you owe a specific amount of money to an entity. For technical and test debt, that is less easy. You can use the quality attributes of tests, technical debt models, and link them to essential indicators. This is a first step. Their technicality and lack of alignment with the

business drivers can make them hard to understand for other people.

Another key factor lies in a cocktail of 3 dangerous factors :

1. Focus on short-term
2. Lack of understanding
3. Delayed feedback loops

Short-sighted approaches come at the cost of piling up debt and side effects in the mid-term. Stakeholders can push in that direction to meet specific objectives, show their power, or just also not being conscious of the trade-offs, the second factor. Our work in quality is to express the associated consequences of specific choices. Not doing that results in the third factor, delayed feedback loops, that are hard for humans to anticipate.

Figure 2: We can act at various levels, but the top of the iceberg is already hard to measure
https://www.intel.es/content/www/es/es/it-management/intel-it-best-practices/reduce-technical-debt-article.html

We can make an analogy with a common situation in human behavior. A child starts to eat some candy. He feels good, then he eats more of them. No one told him about the impacts of candy over time in the body, mind, etc. The children do not care as being young its body processes quickly, in the short-term there is no problem. Over time, he starts to gain various pounds until he

reaches an obese critical state and has to go for surgery.

This is how these 3 factors combined lead organizations to hit a wall, not able to deliver software changes anymore. Then, they have to implement drastic changes if they are still able to survive on the market. Test Debt does not happen overnight. It is the sum of taken and non-taken decisions by a set of actors within the ecosystem.

We identify an essential practice relying on intuition and data, the warning signs.

What are the warning signs of Test Debt?

Test Debt is hard to measure factually, but we can rely on our human capacity to detect, feel and react to warning signs. For test automation, we can sense organizational behaviors and specific test automation attributes.

Let's get back to the Why of our automated tests. One objective of our test automation effort is to accelerate the delivery of software changes with confidence. The test automation value disappears when the team starts to bypass the test automation campaign, search for alternative routes, ask for exceptions. Various reasons are possible as a long execution time, instability, lack of understanding, or other maintainability criteria.

The execution time is directly tied to essential indicators of software delivery: lead-time for changes, cycle-time, and MTTA. These metrics are all part of the Accelerate report, correlating the organization's performance with these measures. We need to constraint our test execution time to limit its impact on these acceleration metrics. For test automation, it means less but more valuable tests executed faster. It can also be about decoupling the tests to be executed under a certain perimeter instead of running each time a large end-to-end regression campaign.

Figure 3: The benefits realized through Test Automation https://www.sogeti.com/explore/reports/world-quality-report-2020/

The second factor of test instability, also known as flakiness, is measured with the flaky ratio. Flakiness is not impacting only the test. It directly impacts the trust of delivering software for your team. It is therefore essential to aim for that percentage of 100%. When this is not the case, understanding and fixing the root causes is a priority. Flakiness can be measured when you have at least one automated test implemented, checking its stability over time. When adding tests, you can also start to measure the overall test suite stability.

The last factor focuses on test automation maintenance warning signs. The key is to act early on specific indicators and behaviors. Regularly assess the impact of the automated test by asking “So what?”. It aims to confirm that the automated tests are having effects on proprietary topics. Are you able to show this type of correlation? If you hear sentences like “I don’t understand what QA is doing anyway”, this is a warning sign. The team can be lost in technical optimizations.

Figure 4: The typical situation of lack of understanding with lack of alignment
https://www.derekhuether.com/blog-details/2013/09/26/lack-shared-understanding

The danger of lacking an alignment is the one of local optimization to the detriment of the whole system. It is dangerous to let a QA team focus on extensive rework, framework optimization, test redesign. They start to take the habit of optimizing their feedback loops without necessarily providing value to the rest of the team. This type of behavior leads to satisfied technical persons completely forgetting about the true goal of their work. Taken too late, you can hear "It's better to start a new framework, it became too complex.". This leads us to poor test design.

Automated tests require design like any proper engineering activity. A number of pitfalls must be avoided, like copying the tests from a manual suite or letting a recorder create tests. You can decide to use some of these assets once the preliminary works of architecture, design and modularization have been completed. Test design is also about selecting which tests to don't implement, an important typology of test to clarify from the start.

The trend of these symptoms is one of unbalance. It is when the costs/benefits equation is not satisfactory anymore.

Smash Test Debt Before It Is Too Late

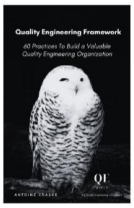
Stakeholders are people that invest money for two reasons, to win or stop losing money. That investment can stop when the test debt comes to an unacceptable gap of actual versus and expected value.

> "Test Debt does not happen overnight. It is the sum of taken and non-taken decisions by a set of actors within the ecosystem."

Antoine Craske

We all want to avoid this situation. Covering the reasons leading to test debt clarifies that the multiple factors are not due to a single person, team, or problem. A more integrated and incremental approach is required.

We cover this approach in the following article about empowering test automation with quality engineering.



Follow the QE Unit for more Quality Engineering right in your inbox

Access the free ebook "60 Practices of Quality Engineeering", get early access to the ebook "On Defining Quality Engineering", the upcoming email serie of the Quality Engineering Framework, exclusive community content, plus a weekly update tailored on Quality Engineering

Email Address *

Follow

Without spam. Read our privacy policy for more info.


The Most Valuable QA Acronyms You Need To Know

How To Empower Test Automation With Quality Engineering

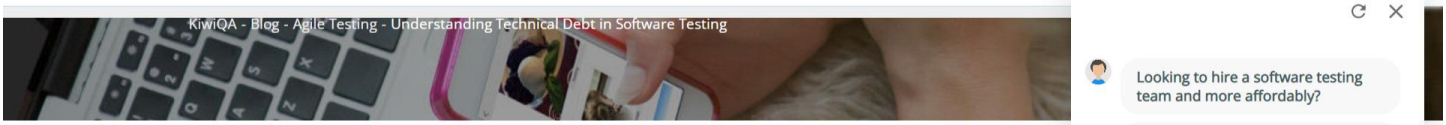

Leave a Reply

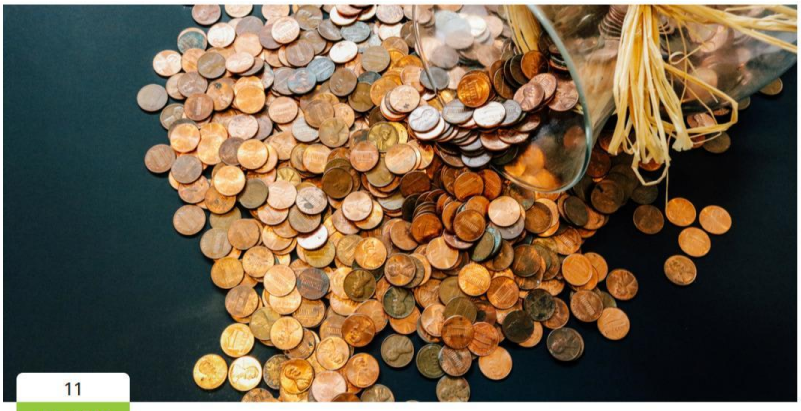
Your email address will not be published. Required fields are marked *

Comment

Name

Email





11
Jun, 2018

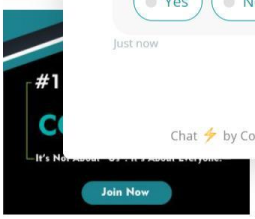
Posted by Niranjan Limbachiya | 0 Comments
| Categories: Agile Testing, Automation Testing, Digital Testing, Software Testing, Test Management

Post Views: 503

Searc

Search he

Join C

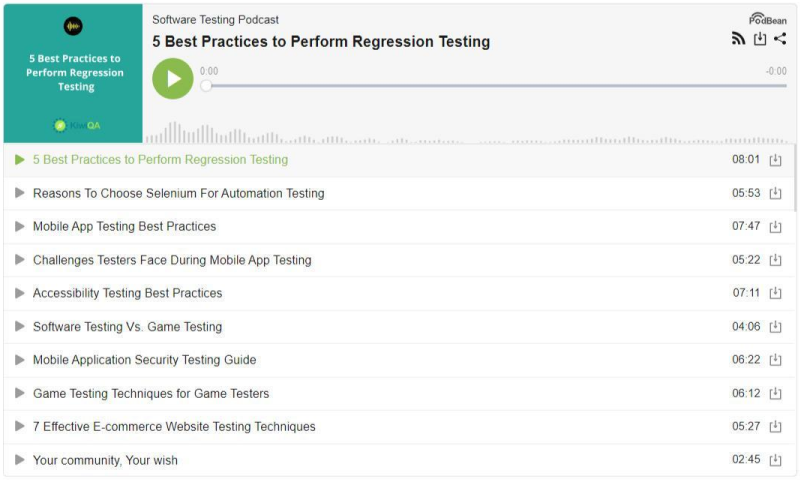


We're

Understanding Technical Debt in Software Testing: Why is it important for Productive Software Testing in Future

Managing technical debt has been one of the most emerging problems for the current general of software testers, especially those organizations that develop and maintain large software systems. Similar to a bad debt in the financial industry, the term was devised by Ward Cunningham to draw an analogy with financial debt to indicate how incurring debt in the short run is beneficial but hurts badly in the long run if not repaid. Basically, the term was meant to remedy the practice of making non-optimal technical decisions.

Technical debt usually occurs when teams knowingly or unknowingly make technical decisions in return for short-term gains in their projects. The test dimension of technical debt is known as 'test technical debt' or 'test debt' for short.



What is Technical Debt?

Ward Cunningham coined the term 'technical debt' wherein long-term software quality is traded for short-term gains in development speed. We incur financial debt when we take a loan. The debt does not cause problems as long as we repay the debt. However, if we do not repay debt, the interest on the debt keeps building over a period of time and finally, we may land in a situation where we are not in a position to repay the accumulated financial debt leading to financial bankruptcy. Technical debt is akin to financial debt. When we take shortcuts during the development process either knowingly or unknowingly, we incur debts. Such debts do not cause problems when we repay the debt by performing refactoring to address the shortcut. If we keep accumulating technical debts without taking steps to repay it, the situation leads the software to "technical bankruptcy".

A high technical debt in a software project impacts the project in multiple ways. If neglected, technical debt can lead to a broad spectrum of difficulties, from collapsed roadmaps to an inability to respond to customer problems in a timely manner. Specifically, the Cost of Change (CoC) increases with increasing technical debt that leads to poor productivity. In addition, it starts the vicious cycle where poor productivity leads to more focus on features (rather than associated quality) in the limited time, and that leads to more technical debts due to time constraints. Apart from technical consequences, high technical debt impacts the morale and motivation of the development teams in a negative manner.

Causes of Technical Debt

There are several well-known causes that lead to technical debt, including:

- Schedule pressures.
- Lack of skilled engineers.
- Lack of documentation.
- Lack of process and standards.
- Lack of test suite.
- Delayed refactoring.
- Lack of knowledge

Test Debt under Manual Testing

In manual testing, test engineers execute test cases by following the steps described in a test specification. Here is a list of common contributors to test debt that relates to manual testing.

- **Limited test execution**– Many a time, the available resources and time to carry out complete and comprehensive testing is not sufficient. Hence, testers execute only a subset of tests for a release (i.e., a shortcut), thereby increasing the possibility of residual defects.
- **Improper test design**– Manual testing is a time consuming and arduous process. A test case could be designed with a lot of variations using different data combinations but then it is necessary for the tester to execute all these combinations. Since the execution of all combination of test cases is a laborious process for testers, they restrict themselves to few happy scenarios for testing taking a shortcut in test design. This increases the risk of residual defects in the System Under Test (SUT).
- **Missing test reviews**– Test reviews help to improve the quality of test cases and also help in finding the problems earlier. Missing test review could delay finding defects or increase maintenance of test cases.

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 infrastructure not resembling customer's infrastructure could lead to unpredictable behaviour of the software resulting in reduced confidence in the system.
- **Lack of coding standards**– Automated tests need to adhere to a common coding standard. When a common coding standard is not adopted and followed, it increases the effort to maintain the test code.
- **Unfixed (broken) tests**– Not fixing broken tests or tests not being updated when functionality changes reduce confidence on the tests, decreases quality and increases maintenance effort.
- **Using record and replay**– It is easy to use record and replay tools for testing Graphical User Interface (GUI) applications. However, using record and replay has numerous drawbacks. Better alternatives may be available to use instead of record and replay tools.

Given the vital role that testing plays in the software development process and the impact test debt can have on software projects, test debt and corresponding management techniques deserve more focus for developing high-quality software in industrial contexts.

Connect with KiwiQA to leverage focused capability for focused QA and Testing services.

Software Testing Podcasts

Tags: manual software testing, Software Testing Debt, Technical debt

ALSO ON KIWIQA'S BLOG

All About CRM Performance...

4 years ago • 2 comments

The aim of any automated CRM Performance Testing tool is to simplify the …

Reasons Why Manual Testing Is Not Dead Yet

2 years ago • 1 comment

Manual testing services are essential in order to examine and eliminate …

Different Stages of Game Testing

4 years ago • 1 comment

Mobile testing can be beneficial for your game before and after its …

Top 10 best Web application Security …

4 years ago • 2 comments

The web testing is of utmost importance in these times due to the rising threat of …

KiwiQA's Blog Comment Policy

We welcome relevant and respectful comments. Off-topic may be removed.

0 Comments KiwiQA's Blog Disqus' Privacy Policy Login

Favorite Tweet Share Sort by Best

Start the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS

Name

Be the first to comment.

Subscribe Add Disqus to your site Do Not Sell My Data DISQUS

Regressi

Reasons
Automat

Mobile A
June 10,

Categ

Select

Archi

Select Month

Tags

Agile (5)
Applica
app tes
automa
big data
biggest
Black bo
cloud ap

Do you need our help?

Yes Not really

Just now

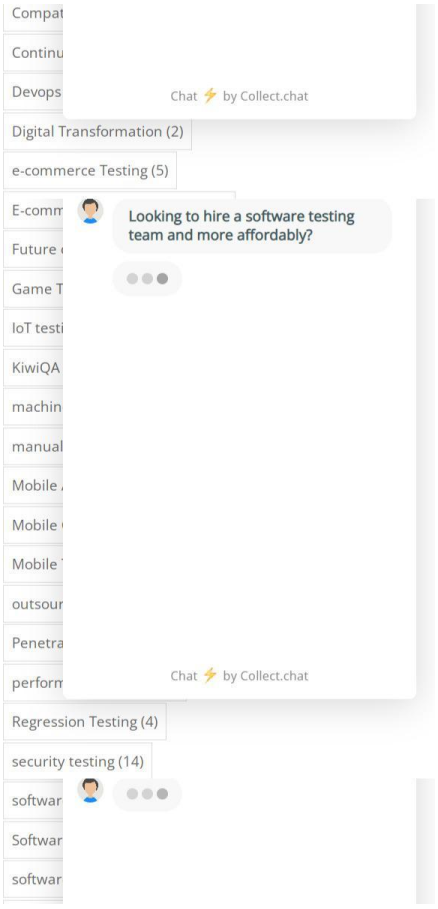
Chat by Collect.chat

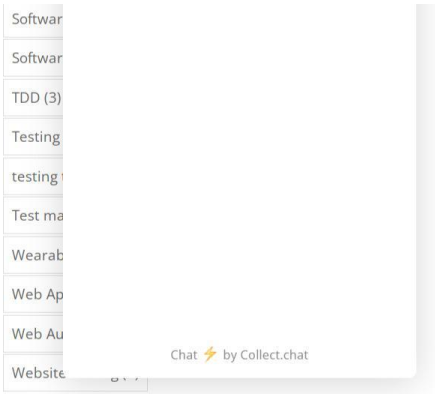
Looking to hire a software testing team and more affordably?

Do you need our help?

Yes Not really

Just now


Compat
Continu
Devops
Chat ⚡ by Collect.chat
Digital Transformation (2)
e-commerce Testing (5)
E-comm
Looking to hire a software testing team and more affordably?
Future
Game T
IoT testi
KiwiQA
machin
manual
Mobile
Mobile
Mobile
outsour
Penetra
perform
Chat ⚡ by Collect.chat
Regression Testing (4)
security testing (14)
softwar
Softwar
softwar



Information

- Company
- Contact Us
- Career
- Blog
- Videos
- Privacy Policy
- Legal Notice

Our Expertise

- Automation Testing Services
- AI Testing Services
- Performance Testing
- Mobile Testing Services
- Security Testing Services
- Game Testing Services
- Manual Testing Services
- Test Consulting

Digital Assurance

- CRM Testing Services
- Game Testing Services
- SalesForce Testing
- E-commerce Testing
- Big Data Testing
- Company Broucher
- Corporate Presentation
- Software Testing Training

KiwiQ

BACK TO BLOG

Test debt explained. Why your team has no spare test capacity.

Test debt is a real problem for most QA teams. It eats your capacity and results in less and less productive testing being done as I explain here

JULY 7, 2021 TAMAS CSER

General

Many years back, I realize
automation. Not only were the tools woefully outdated, they also were driving teams into test debt. Here, I explain what

Similar posts

Implementing AI Solutions in a Down Economy

> this means and show how it is really impacting your company. I've also worked with my team to create a useful infographic that illustrates the problem. - Tamas Cser

The problem with test automation

Test automation is essential to keep up with the pace of modern application development. However, over the years, test automation tools have failed to keep up. They are stuck in the dark ages compared with the tools your development team uses. The problem is, most legacy test automation relies on test scripting. But test scripts are notoriously prone to breaking as your application changes. The result is, your team starts to spend more and more time on test maintenance and less and less on creating tests. In turn, they lose focus on their central aim, which is ensuring your software is bug-free and reliable. I call this problem test debt.

Test capacity

Every team has a finite capacity they can spend on test automation. They have to split this time between creating tests and analysing test results. When a team starts automating tests, they have plenty of spare test capacity. But that rapidly reduces as they become familiar with the tools and start to automate more tests. More automated tests means more test results to analyse. Eventually, they will reach their overall capacity. This idealized case is shown in the graph below.

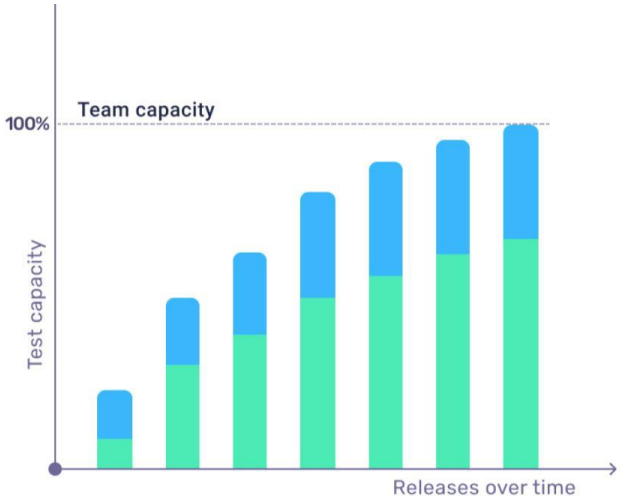

What is Parallel Execution?


Smoke vs. Regression Tests


Why Shorter Tests Are Better


7 Principles of Software Testing


Top Book Recommendations For Software Testing Leaders

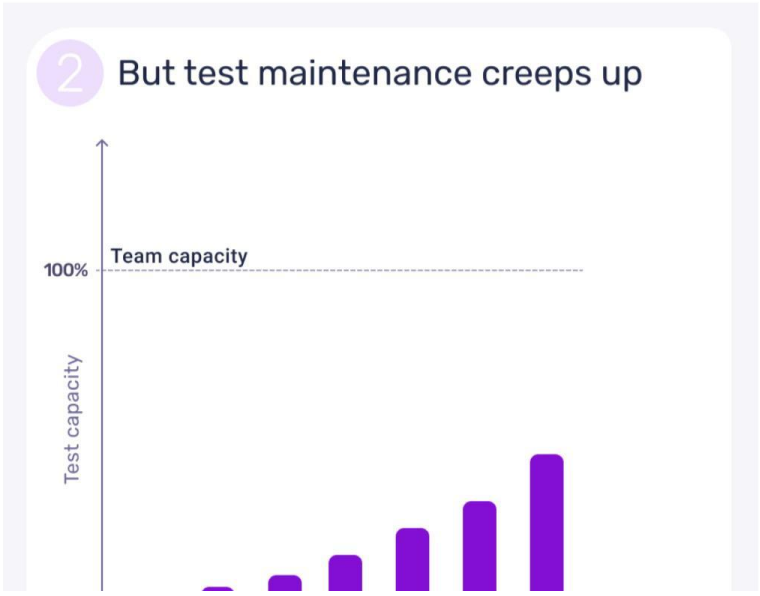


Managers think QA teams split their time between test creation (green) and test analysis (blue)

Test maintenance

So, all seems well with your QA team. They are automating more tests and increasing

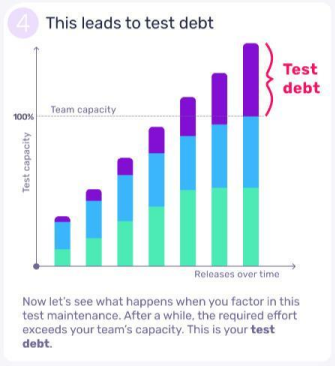
So, all seems well with your QA team. They are automating more tests and increasing your test coverage. But there's a fly in the ointment. Test scripts rely on selectors (like element ID or xpath) to find elements in your UI. They can then interact with that element and check what happens. For instance, you can tell the script to click a specific button or enter text in a certain field. The problem is, these selectors change unpredictably as your application changes. Even simple styling or layout changes can result in the script choosing the wrong button, or entering text in the wrong field. The upshot is, when you update your app, your tests will suddenly fail. In a few cases, there may be a genuine bug. But more often than not, the test just needs to be fixed. As they automate more tests, they spend more and more time on this test maintenance, as shown in the graph below.

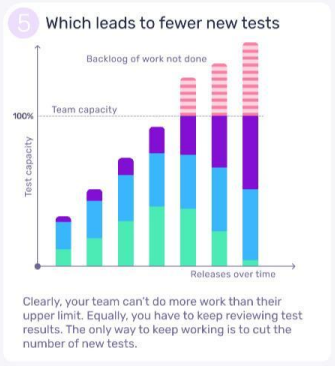


But actually, as your application develops, the team has to spend an increasing share of time on test maintenance—often we see teams who spend 50% of their time on this.

Factoring in maintenance

Eventually, the team can end up spending half their time on test maintenance. If you now add this on to the idealised graph, you can see you rapidly hit a problem. The overall work required significantly exceeds their capacity. This is known as test debt. Teams that are in test debt are going to really struggle badly.





When you take test maintenance into account, you actually see a backlog of testing

The impact of test debt

Test debt is one of the most damaging problems your QA team faces. When you are in test debt, you have to make some really tough decisions. You are stuck between a rock and a hard place and need to sacrifice existing maintenance or new coverage:

- Focus your attention on fixing the broken tests and just rely on manual testing for all your new features. That works OK if you have spare manual test capacity. The problem is, it's probably not sustainable, since every new feature will break more and more tests.
- Continue to increase test coverage and ignore some of the broken tests. That is a massive gamble though, since there was a good reason to automate those tests!

Worse still, you eventually start to see your test coverage decline, as the graph on the right shows. This happens because as you develop new features you have to keep developing new tests. But your team can't automate these new tests while they're fixing existing tests. So the team grows dependent on manual testing, which is another huge time sink.

Eventually, QA teams find they have become the roadblock to releasing new features. Seeing this problem made me realise that there has to be a better way to do things. To me, the obvious answer was to use machine learning to try and reduce the need for test maintenance. And so I founded Functionize.

How Functionize solves test debt

Functionize has been an AI company from day one. Our approach has been to apply machine learning and other AI approaches to solve the problems with test debt. Specifically, we tackle it in three ways:

- **Reducing test maintenance**. Every time you run a test on the Functionize platform, it records a vast amount of data. This includes details of objects on the page, hidden objects, calculated CSS, API calls, timings, and more. We use this data to create a detailed ML model of your entire application. In turn, this allows us to use Smart Element Recognition to avoid the need for routine maintenance. In effect, our model works out what changed in the UI just like your manual testers would. If a "Buy now" button moves, changes style and gets changed to "Add to cart" a human still knows it's the same button. Well, so does our system. That means many fewer tests fail just because your UI was updated.
- **Speeding up test creation**. The other way to increase test coverage is to reduce the time needed to create each test. This will mean your team can use the same test capacity to create more tests. Our Architect smart recorder allows you to create AI-powered tests in just minutes. The tests automatically work on any browser and platform. By contrast, creating a test script can take hours, and then it has to be refined for each platform or browser you want to test. All this is only possible because we combine different AI approaches including deep learning, natural language processing, and computer vision.
- **Simplifying test analysis**. One of the best things with AI is the ability to apply techniques like computer vision. This allows us to simplify test analysis no end through our visual testing approach. We record screenshots for every test step and every single test run. We are able to use these to highlight parts of your UI that have changed more than expected since the last run. You can also choose to validate specific parts of the screen (for instance, your logo). Alternatively, you can do an in-depth visual verification of the entire UI, including CSS as well as visual elements. Overall, this streamlines test analysis and makes it accessible to everyone.

With Functionize

Functionize solves test debt, and brings many other benefits that boost QA productivity

Overall, Functionize allows you to cut test debt, increase test coverage and keep your team focused on delivering better products, faster. All this is explained on our new infographic - download it now.

Popular posts

6 ways to improve your debugging skills

27 Selective QA Interview Questions for Managers to Ask

So you want to become a software QA professional?

A glossary of testing terms

Alpha testing vs Beta testing. Why they matter for you

Software testability – what it it is and how to improve it

What Is Gherkin + How Do You Write Gherkin Tests?

The Day to Day Activities of A QA Tester

Categories

AI Visual Testing | API Testing | Agile QA | Artificial intelligence | Autonomous Testing | CI/CD | Canary Testing | Career | Cloud Testing | Continuous Quality | Continuous Testing | Cross Browser Testing | Data driven testing | DevOps | Digital Transformation | End-to-end testing | Functionize | General | Load Testing | Localization testing | Low Code/ No Code | Management | Mobile Testing | PR | Performance Testing | Product Releases | Selenium | Software quality | Test Creation | Test Ops | Test maintenance | Testing | User Acceptance Testing | case study | shift-left | shift-right

Automated Testing

United States Patent and Trademark Office (USPTO)

USPTO OFFICIAL NOTICE

Office Action (Official Letter) has issued
on August 23, 2022 for
U.S. Trademark Application Serial No. 97120746

A USPTO examining attorney has reviewed your trademark application and issued an Office action. You must respond to this Office action in order to avoid your application abandoning. Follow the steps below.

(1) Read the Office action. This email is NOT the Office action.

(2) Respond to the Office action by the deadline using the Trademark Electronic Application System (TEAS). Your response must be received by the USPTO on or before 11:59 p.m. **Eastern Time** of the last day of the response period. Otherwise, your application will be abandoned. See the Office action itself regarding how to respond.

(3) Direct general questions about using USPTO electronic forms, the USPTO website, the application process, the status of your application, and whether there are outstanding deadlines to the Trademark Assistance Center (TAC).

After reading the Office action, address any question(s) regarding the specific content to the USPTO examining attorney identified in the Office action.

GENERAL GUIDANCE

- **Check the status of your application periodically** in the Trademark Status & Document Retrieval (TSDR) database to avoid missing critical deadlines.

- **Update your correspondence email address** to ensure you receive important USPTO notices about your application.

- **Beware of trademark-related scams**. Protect yourself from people and companies that may try to take financial advantage of you. Private companies may call you and pretend to be the USPTO or may send you communications that resemble official USPTO documents to trick you. We will never request your credit card number or social security number over the phone. And all official USPTO correspondence will only be emailed from the domain "@uspto.gov." Verify the correspondence originated from us by using your Serial Number in our database, TSDR, to confirm that it appears under the "Documents" tab, or contact the Trademark Assistance Center.

- **Hiring a U.S.-licensed attorney.** If you do not have an attorney and are not required to have one under the trademark rules, we encourage you to hire a U.S.-licensed attorney specializing in trademark law to help guide you through the registration process. The USPTO examining attorney is not your attorney and cannot give you legal advice, but rather works for and represents the USPTO in trademark matters.

Note To The File

Serial Number: 97120746 TEST DEBT
Date: 08/23/2022 6:54 pm
Created by: Kimberly Parks

Searched

- Google
- OneLook
- applicant's website

User: Kimberly Parks

Statistics for Case 97120746						
#	**Search**	**Total Marks**	**Dead Marks**	**Live Viewed Docs**	**Live Viewed Images**	**Status/Search Duration**
1	97120746[sn]	1	0	0	0	0:00
2	*te{"sz"}t*[bi,ti]not dead[ld]	4647	0	0	0	0:01
3	*debt*[bi,ti]not dead[ld]	336	0	0	0	0:02
4	2 and 3	1	0	1	1	0:00
5	3 and ("042")[cc]	291	0	0	0	0:00
6	3 and (a b "042" "200")[ic]	36	0	36	36	0:00
7	3 and (a b "009" "200")[ic] not 6	30	0	30	30	0:01
8	2 and (a b "009" "200")[ic] not 6	930	0	0	0	0:02
9	2 and (a b "042" "200")[ic] not 6	942	0	0	0	0:01

Session started 08/23/2022 5:49 pm
Session ended 08/23/2022 6:52 pm
Total search duration 7.00
Session duration 1 hours 3 minutes 5 seconds
Adjacency Level 1
Near Level 1