

# Proceedings

**42nd Annual  
Pacific Northwest Software Quality Conference**

**a Hybrid Conference  
in Portland, Oregon and via Zoom Teleconference**

**October 14-16, 2024**



*Permission to copy without fee all or part of this material, except copyrighted material as noted, is granted provided that the copies are not made or distributed for commercial use.*

# Table of Contents

	<i>Page</i>
<b>Forward</b>	5
<b>Conference Papers</b>	
<i>in alphabetical order of first author's surname</i>	
1. Quality Beyond Testing with DevOps: Jenkins to GitHub Actions - Migration for Enhanced Process Optimization and Quality Assurance - by Sagar <b>Aghera</b> , Nikhil Yogesh, and Joshi Amit Bhanushali	9
2. AI-Driven Techniques for Noise Filtration in Software Validation Logs - by Tarun <b>Arora</b>	19
3. Shifting Other Than Left or Right - by Jon <b>Bach</b>	28
4. Artificial Intelligence & Machine Learning in Test Data Management - by Praveen <b>Bagare</b>	36
5. Product Owner in Test - by Sridevi Sabarinath <b>Balla</b>	44
6. Achieving Software Quality Excellence Through Software-Hardware Co-Design and Co-Verification - by Monica <b>Bao</b> , Sophia Lee, Havish Sripana, and Joanna Mei	54
7. We need to talk...(more) about Mental Health at Work - by Veronika <b>Betzel</b>	66
8. QA Authority: From Zero to Hero - by Anton <b>Bolotin</b>	76
9. The Future is Finding Zero Bugs (in QA) - by Leslie <b>Brooks</b>	88
10. Team Leadership in the time of remote silence - by Michal <b>Buczko</b>	98
11. SQA skillset, AI and Advancing medical research - by Tracy T. <b>Burridge</b>	112
12. Enhancing Validation Through Attestations - by Brent <b>Clausner</b>	124
13. Greater Quality Through Tactical Domain-Driven Design - by John R. <b>Connolly</b>	137
14. Vulnerabilities in Deep Learning Language Models: Security Risks and Mitigation in Non-Federated, Federated and Decentralized Training - by John <b>Cvetko</b> and Bhushan Gupta	149

15.	Mutating Test Scenarios: Explore the unknown by disregarding order and welcoming events - by Baubak <b>Gandomi</b>	160
16.	Inside Scoop on Risk-Based Testing - by Robin F. <b>Goldsmith</b>	173
17.	Revolutionizing Test Automation:Harnessing Generative AI to Empower Product Management and Accelerate SDLC - by Artem <b>Golubev</b>	185
18.	IoT in the Matrix: Mocking External Dependencies for IoT Systems Using Physical Hardware - by Bartholomew <b>Hsu</b> , Jovan Araiza, and Jeff Borg	195
19.	Trust by Design: Building Credibility in AI with User-Centered Principles - by Zahra <b>Langford</b>	206
20.	Scaling Test Automation using a Metadata-centric Architecture - by Donald <b>Maffly</b>	217
21.	Bringing the Dawn of Security & Compliance to Project Teams - by Vivek Sahai <b>Mathur</b>	226
22.	The Test SDK: The SDET's Best Friend in Monolith-to-Microservices Migrations - by Wayne <b>Ng</b>	236
23.	Exceeding Expectations, Inspiring Quality - by Nishadhi <b>Nikalandawatte</b> and Andy Jimenez	248
24.	Scaling Technical Work While Maintaining Quality - Leveraging Community Principles - by Katherine <b>Payson</b>	262
25.	Fusing Shift Left Testing, AI Solutions & Automation to Elevating Testing Efficiency - by Peh Wei Wooi and Ang Kok Cheang	273
26.	Secure Software, Inside and Out: Seven Steps - by Joye <b>Purser</b> , Jan Graves, and Walter Angerer	289
27.	Quality Maturity Model: Driving Excellence in High-Growth Orgs - by Vidhya <b>Ranganathan</b>	294
28.	Enhance Software Performance Testing with Artificial Intelligence - by Rini Susan V S	309
29.	Signals-based Testing of Microsoft Office with an AI-Driven Robot Army - by Wayne <b>Roseberry</b>	317

30.	Integrating Generative AI for Quality Engineering into the Software Lifecycle - by A. <b>Rozumenko</b> , A. Udovychenko, and T. M. King	330
31.	What would you say you do here? Reframing your Role in the QA Field - by Jeff <b>Sing</b>	348
32.	Software and Methods Supporting Legal Proceedings: Home-Brewed and Speculative or Verifiably Reliable and Definitive? - by Kalman C. <b>Toth</b>	360
33.	Leveraging Wiring the Winning Organization for Organizational Transformation - by Ron <b>Wilson</b>	372
34.	Turning Testers into Superheroes - by Jonathon <b>Wright</b>	390
35.	Data Quality Assurance in Credit Card Fraud Prevention - by Yuqing <b>Yao</b>	401

# Forward

This is the 42nd anniversary of the Pacific Northwest Software Quality Conference (PNSQC). We celebrate another great year by acknowledging the entire PNSQC community: our attendees, authors / presenters, peer reviewers, volunteers, sponsors, and board members. Thank you for your continued support!

You will continue to find PNSQC to be one of the best conferences in terms of learning state-of-the-art and practical methods for improving software quality. PNSQC 2024 continues this tradition.

This year's conference featured speakers from around the globe in four tracks. During the three days of PNSQC 2024, our technical program contains keynote addresses, invited presentations, peer-reviewed papers & presentations, panel discussions, tutorials, workshops, and exhibits of our sponsors. For additional details, see [www.pnsc.org](http://www.pnsc.org).

PNSQC is held using a hybrid format; with a physical venue in Portland, Oregon and a simultaneous virtual venue via Zoom. We are pleased that close to 200 attendees participated in person or virtually. The majority of the attendees came to Portland in-person, at a level exceeding the Board's expectation. This reflects a strong and vibrant PNSQC community.

This volume contains all papers that successfully went through the PNSQC peer review process. In this process, at least two reviewers commented on each paper for possible improvement by its authors – typically over three drafts. This approach assures quality of our contributed papers and is an important differentiation between PNSQC and other industry conferences on software quality. We hope you find these materials informative and useful.

Best Regards,

Ying Ki Kwong, PhD  
Board Member  
PNSQC

This page is intentionally left blank.

## **Conference Papers**

This page is intentionally left blank.

# Quality Beyond Testing with DevOps: Jenkins to GitHub Actions Migration for Enhanced Process Optimization and Quality Assurance

**Sagar Aghera      Nikhil Yogesh      Joshi Amit Bhanushali**

[saghera@netskope.com](mailto:saghera@netskope.com) -- [nikhil.yogesh.joshi@fiserv.com](mailto:nikhil.yogesh.joshi@fiserv.com) -- [amit.bhanushali@mail.wvu.edu](mailto:amit.bhanushali@mail.wvu.edu)

## Abstract

DevOps integrates and automates software development (Dev) with IT operations (Ops), complementing Agile SDLC by using automation tools to speed up software delivery. The choice of Continuous Integration/Continuous Deployment (CI/CD) tools is crucial for streamlining these processes. This paper presents our organization's migration from Jenkins to GitHub Actions for better scalability, security, and integration.

Previously, our organization faced challenges with Jenkins regarding scalability, security vulnerabilities, and complex third-party integrations. GitHub Actions provided a comprehensive platform for automating tasks within GitHub repositories. Our DevOps team automated processes like Pull Request checks, nightly builds, code coverage scans, and test executions. Integrations with JFrog Artifactory, Slack, Red Hat Ansible, and Atlassian JIRA facilitated a smooth transition.

GitHub Actions' secret management eliminated the need for additional tools like HashiCorp Vault and 1Password, enhancing security and simplifying secret storage. Its multi-language support, compatibility with major operating systems, and built-in secret management ensured a seamless migration. This paper discusses the challenges, benefits, and key considerations of the migration process, demonstrating how GitHub Actions improved our automation, workflow efficiency, and code quality assurance in a modern DevOps environment. After migrating to GitHub Actions, developer efficiency increased by 35%.

Index Terms—DevOps, Automation testing, CI, CD, Jenkins, GitHub Actions.

## Biography

Sagar Aghera holds a master's in computer science and a bachelor in electrical engineering from Florida Atlantic University, with over 10 years of experience in DevOps, CI/CD & test automation. He has worked at Motorola, Qualcomm, Gracenote, VMware, and currently excels as a Sr Staff Software Engineer at Netskope, focusing on testing frameworks, cloud infrastructure, and mentoring. His commitment to Testing and DevOps is evident throughout his career.

Nikhil Yogesh Joshi, a Senior Manager in Software Engineering based in Cumming, Georgia, boasts over 18 years of experience leading high-performing teams and delivering complex projects. His career spans multiple industries, demonstrating his expertise in automation, cloud migration, and strategic leadership.

Amit Bhanushali is a Manager, Software Quality Assurance at West Virginia University based in West Virginia, United States and have more than 20 years of experience in BFSI and Higher Ed Domains. He is a certified Scrum Master and holds a master's degree in business data Analytics from West Virginia University in West Virginia, United States.

# 1 Introduction

## 1.1 Background

DevOps has become an increasingly important practice for organizations that need to keep up with the demands for rapid development and testing. In today's evolving software development landscape embracing DevOps practices has become crucial for companies looking to improve teamwork, between their development and operations teams. This shift in approach aims to boost how often code is deployed and enhance the quality of services. Contrary to popular belief, DevOps does not replace QA participation or oversight requirements. However, It will necessitate a change in how QA plans and does testing in a DevOps environment. DevOps quality assurance will require a considerable cultural shift away from traditional testing methods and more development, operational, and QA team collaboration. In an earlier study, the effectiveness of implementing DevOps practices in enhancing continuous delivery has been demonstrated (Sharma 2018).

DevOps transcends mere tool adoption; it represents a transformation that demands a reexamination of testing methods. This article delves into QAs evolving role in the age of DevOps focusing on the strategies, obstacles and future paths for quality assurance testing. Continuous testing emerges as vital in this scenario by weaving testing activities into every stage of development. As businesses strive to deliver top notch software the importance of QA approaches grows more evident. Continuous testing delivers continuous quality for any integrated CI/CD pipeline.

Traditionally the divide between Dev, QA & Ops teams – often termed as the "Wall of Confusion" – impeded communication and collaboration. Wall of Confusion here stands for Separation Of Duties (SOD). People often use this wall to get out undesirable tasks. To break this wall, DevOps practices were introduced. One can promptly resolve many QA issues in a DevOps environment by continuous improvement in communication and collaboration. The QA and development teams have traditionally operated in isolated groups, thus it's vital to initiate communication and ongoing feedback between them. DevOps seeks to break down this barrier by fostering an ethos towards software delivery. This approach stresses accountability, for service excellence with ongoing testing serving as a foundation. This provides an opportunity for QA to add an invaluable skill set in their repertoire.

## 1.2 Objectives

This article delves into the foundation of DevOps discussing its principles, advantages and tools particularly Jenkins and GitHub Actions which support CI/CD processes. Jenkins, an open-source automation server or platform has been widely employed to automate tasks but it does come with challenges such as maintenance overhead and security issues. Jenkins can be challenging to set up and maintain with outdated UI & requires regular upkeep & maintenance. It also demands careful configuration to ensure security, especially in public environments. On the other hand, GitHub Actions offers a streamlined approach by seamlessly integrating with GitHub repositories and providing robust support for automating workflows. GitHub Actions provides a user-friendly platform for developers & testers. GitHub Actions removes the necessity of managing servers, which leads to a decrease in operational expenses. GitHub Actions can run secluded jobs in secure environments, which come along with modern safety features like secrets management, access control, etc.

The shift from Jenkins to GitHub Actions marks an advancement in optimizing QA & DevOps procedures. This shift addresses issues linked to Jenkins like scalability problems and plugin dependencies while boosting efficiency and collaboration. The migration process involves planning, testing and validation to ensure a transition that upholds quality & DevOps standards.

Through case studies and practical applications this article showcases the advantages of embracing GitHub Actions for DevOps & QA purposes. These benefits include streamlined CI/CD workflows, quicker execution times and enhanced collaboration while making no compromises on security. The transition not only cuts down expenses but also elevates the overall reliability and security of the software delivery pipeline.

In summary, incorporating DevOps methodologies into quality assurance is crucial for upholding top notch standards in a changing software development landscape. With the increasing adoption of DevOps

by companies QA approaches need to adapt to address emerging obstacles and make use of cutting edge tools and practices. This document offers perspectives on the tactics and upcoming trends in QA during the DevOps era highlighting the significance of enhancements and teamwork.

## 2 Understanding Jenkins

### 2.1 Overview

As per Jenkins documentation, Jenkins is a self-contained, open-source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software (Jenkins Documentation). It helps automate the parts of software development related to building, testing, and deploying, facilitating DevOps practices by allowing developers to integrate changes into a shared repository frequently and obtain rapid feedback. According to the Jenkins Documentation, the pipeline features enable automation of software deployment processes effectively (Jenkins Documentation).

### 2.2 Strengths & weaknesses

Jenkins provides a high level of customization due to its open-source design, range of plugins and ability to integrate with different CI/CD systems. However users need to invest time and effort in configuring and maintaining it properly to deal with performance challenges, plugin compatibility issues and security concerns, which can be demanding and intricate tasks. Table.1 highlights a few strengths and weaknesses.

Strengths	Weaknesses
<b>Open Source</b>	<b>Complex Configuration</b>
Free to use and supported by a large community.	Can be complex to set up and configure, especially for beginners.
<b>Flexible and Extensible</b>	<b>Maintenance Overhead</b>
Highly customizable to fit various CI/CD workflows.	Requires regular maintenance and updates to ensure stability and security.
<b>Active Community and Documentation</b>	<b>User Interface</b>
Large, active community providing support and comprehensive documentation.	The UI can be seen as outdated and not as user friendly as other CI/CD tools.
<b>Cross-Platform</b>	<b>Security</b>
Runs on various operating systems including Windows, macOS, and Linux.	Requires careful configuration to ensure security, especially in public-facing environments.

Table.1 Jenkins strengths vs weaknesses.

## 3 Introduction of GitHub Actions

As described in GitHub Actions documentation, GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment

pipeline. You can create workflows that build and test every pull request to your repository or deploy merged pull requests to production (GitHub Actions: Overview). It is an elegant solution that offers the convenience of creating and managing CI and CD workflows from GitHub, the largest code-hosting platform in the world (Heller 2021). There is a rich ecosystem of more than 20,000 3<sup>rd</sup> party actions, for use. Fig.1 highlights the key components of GitHub Actions. GitHub Actions Marketplace provides all the necessary integration points for most of the widely used tools & platforms such as slack, Azure, AWS, JIRA, Artifactory etc.



Fig.1 Key components of GitHub Actions.

## 4 Benefits of migrating to GitHub Actions

GitHub is by far the largest social coding platform, hosting the development history of millions of collaborative software repositories, and accommodating over 100 million users by January 2023 (100 Million Developers and Counting, 2023). GitHub Actions integrates closely with GitHub providing a safe and budget friendly CI/CD experience supported by a range of marketplace options. It boosts developer efficiency with an easy to use interface and adaptable workflows. Table.2 sheds light on key capabilities of GitHub Actions.

Capability	Description
Integration with GitHub	GitHub Actions is tightly coupled with the GitHub platform offering a unified workflow experience. This close integration enables Devs & QA to automate, personalize, and oversee their software development cycle within the GitHub repository.
Enhanced Developer Experience	GitHub Actions provides a user-friendly platform for developers & testers. By using YAML syntax to outline workflows, it becomes simple and approachable. Teams have the option to utilize actions from the GitHub Marketplace or design personalized actions tailored to their requirements.
Cost and Resource Efficiency	Using GitHub Actions removes the necessity of managing CI/CD servers, which leads to a decrease in operational expenses. It offers solutions that expand alongside the project, guaranteeing resource utilization.
Reliability & Security	GitHub Actions can run isolated jobs in secure environments, which come along with modern safety features like secrets management, access control, etc. It makes it possible to preserve sensitive data and information.

Capability	Description
Ecosystem and Marketplace	The GitHub Market provides a rich ecosystem of pre-built actions created by the community and industry leaders and curated by GitHub. This extensive library, comprising more than 20,000 actions, allows developers to easily integrate 3rd party tools into their workflows.

Table.2 GitHub Actions capabilities

## 5 Migration Process: Step-by-Step Guide

GitHub does provide Actions importer tool to facilitate migration from widely used tools like Azure DevOps, Bamboo, Jenkins, CircleCI and BitBucket but we didn't use this tool for our migration. As per Rautainen, the migration still requires a lot of manual transformation and verification especially when custom scripts and nested and conditional steps are involved (Rautainen, 2023). Migrate from Jenkins to GitHub Actions by analyzing the current setup, planning the transition, converting pipelines to GitHub workflows, testing in staging, and deploying to production while monitoring performance, as presented in Fig.2 and discussed in below sub-sections.

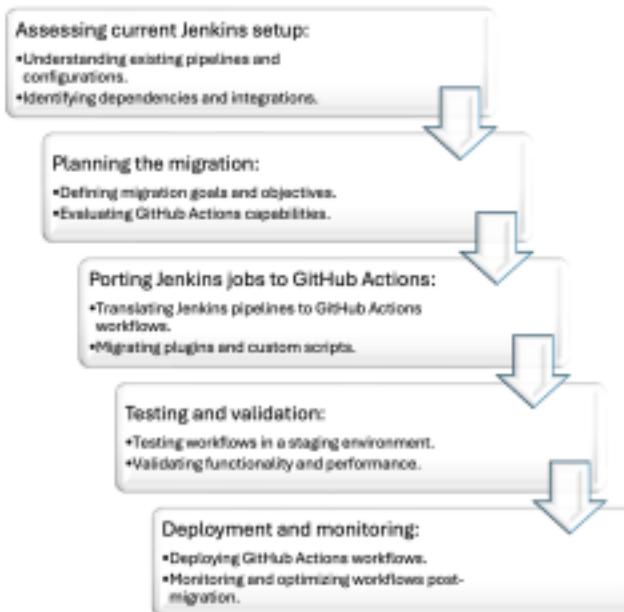


Fig.2 Migration process

For a smooth migration, the first step is to methodically analyze the current Jenkins setup which includes pipelines, freestyle jobs and configurations. Document it and additionally, identify any external plugins and custom scripts. These dependencies will need to be addressed during the migration. The second step is to plan the migration by identifying goals and objectives of migrating to the GitHub Actions. Mostly, it has to do with understanding of capabilities, features, and understanding of workflow syntax and available actions.

In the third step, the Jenkins job and pipeline are ported to GitHub action workflows. This will involve

translating the Groovy based pipeline syntax to the YAML based GitHub Actions workflow syntax. Minor adjustment of the logic or functionality might be needed to align with GitHub Actions features. For 3<sup>rd</sup> party plugins or custom scripts used in Jenkins, identify equivalent GitHub Actions built-in actions or 3rd party actions. One of the most frequently made mistakes is trying to migrate workflows in as-is status, the ability of GitHub Actions may allow us to simplify our workflows and reduce complexity, this often causes a new user to go down a rabbit-hole and waste time & energy. For example, in one of the case studies discussed in the next section, we were migrating a custom script to create tag and cut a release which was readily available as a built-in action.

At last, setting up the staging environment to test the GitHub Actions workflows meticulously. This will help in identifying any issues before we deploy to the production environment. Conduct comprehensive testing, including edge cases & failure scenarios to make sure workflows are robust enough. And if the performance gain was a criterion for migration, then make sure migrated workflows meet expectations. Once testing & validation is done, deploy the GitHub Actions to the production environment. Continue to monitor the performance & behavior of workflows and identify any areas of improvement if needed.

To further emphasize the key benefits of above migration practices, we compiled our results from 3 case studies which are described in the next section.

## 6 Case Studies

### 6.1 Case Study 1: Client builds pipeline

Netskope encountered difficulties, with their Jenkins build process, for Endpoint DLP client product. The challenges involved upkeep, intricate setup handling and prolonged build durations all of which impeded engineering productivity and caused release delays at times. The approach included transitioning from Jenkins to GitHub Actions to simplify and modernize the build pipeline. GitHub Actions was chosen for its compatibility with GitHub repositories, easy setup and robust automation features. Table.3 covers the pipeline steps involved when using Jenkins and after migrating to GitHub Actions. One notable advantage we had with migration is use of readily available Built-in action to tag the build and release which was done using custom script written & maintained by DevOps team. This resulted in per build time reduction from 50 mins in Jenkins to about 40 mins in GitHub Actions which is about 20% reduction in execution time. Additionally, we had more stable runs after migration, with failure rate reduction from 15% to 3%.

Pipeline Steps	Jenkins	GitHub Actions
Source code checkout	GitHub plugin	Built-in Action
Build product	Built-in shell plugin	Built-in Action
Run unit tests	Built-in shell plugin	Built-in Action
Upload artifacts	JFrog plugin	JFrog Action
Tag build & release	Custom script	Built-in Action
Send email	Email-ext post build action	Send Email Action
Slack notification	Slack Notification plugin	Slack API Action

Table.3 Pipeline steps – Jenkins vs GitHub Actions

## 6.2 Case Study 2: Pull request sanity

Given the Jenkins infrastructure and GitHub being in a separate network, Jenkins hosted in a local datacenter, webhook to GitHub from Jenkins server was not configurable. Pull request sanity check was a new addition to CI, given it was readily available with GitHub Actions. GitHub Actions is event driven and one of the supported events is Pull request. This made the creation of pull request sanity check quite trivial and many of the steps from client build pipeline in previous case study were reused. Fig.3 covers the steps involved in Pull request sanity check. This PR sanity check was more effective because it allowed us to isolate issues earlier, which would help in stopping issues entering the main branch. Additionally, code coverage check helped in following TDD (Test Driven Development) principles.

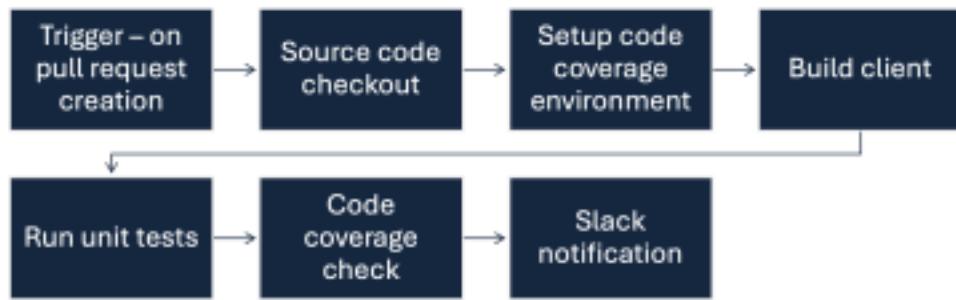


Fig.3 Pull request sanity check steps

## 6.3 Case Study 3: Test automation run(s)

Before migration, test automation runs were done using Jenkins freestyle projects. There was usage of 3<sup>rd</sup> party plugins – source code checkout, test report parser, custom email plugin, and slack notification plugin. Migrating to GitHub Actions involved matching with equivalent Actions and creating a workflow file with the steps calling those actions. One notable point here, the resultant workflow file contained less than 75 lines. Template of the workflow file in use is shown in Fig.4

The workflow had a trigger to run on schedule (every weekday). The workflow consisted on one job with below steps in order,

1. Automation source code checkout
2. Run python tests
3. Parse the junit report
4. Post the results to slack channel

```

1   name: EPDLP Test Automation
2   on:
3     schedule:
4       - name: run smoke tests
5         cron: "@0 12 * * 1-5" # run every week day
6
7   jobs:
8     run_smoke_tests:
9       runs-on: [self-hosted, linux, x64, epdlp-qel] # Tests are run on self-hosted Linux VM.
10      steps:
11        - name: Automation Checkout # Automation source code checkout
12          uses: actions/checkout@v3.1.0
13
14        - name: run PDVs # Running Python based test automation
15          run: |
16            | pytest -m "smoke" --junitxml=smoke_tests.xml
17
18        - name: Test Report # Parsing Junit report
19          id: test-results
20          uses: EnricoMi/publish-unit-test-result-action@v2
21          with:
22            action_fail: true
23            files: |
24              | ./*.xml
25
26        - name: Post to a Slack channel # Posting test outcome to slack channel
27          uses: slackapi/slack-github-action@v1.24.0
28          if: success() || failure()
29          with:
30            channel-id: <channel-id>
31            slack-message: "Test outcome: ${fromJSON(steps.test-results.outputs.json).conclusion}\nTotal tests: ${fromJSON(steps.test-results.outputs.json).stats.tests}\n${fromJSON(steps.test-results.outputs.json).stats.tests_succ} passed\n${fromJSON(steps.test-results.outputs.json).stats.tests_fail} failed"
32            env:
33              SLACK_BOT_TOKEN: ${secrets.SLACK_TOKEN}
34
35
36
37
38

```

Fig.4 Template of workflow file for test automation run(s)

## 6.4 Results and outcome

According to Smith, GitHub Actions significantly enhance the efficiency of workflow automation in development environments (Smith, 2020). The tight integration of GitHub Actions with GitHub streamlines the entire workflow process, and hence eliminating the need to manage separate systems for version control and CI/CD. By eliminating the costs associated with infrastructure maintenance, scaling, and support, about 15% reduction in operational costs was achieved after migrating from Jenkins to GitHub Actions. Developer productivity increased by about 10% due to tight integration and less context switching. After migrating to GitHub Actions, developers & QA can manage their code repositories, monitor CI/CD & automation runs all within the same platform and hence eliminate context-switching between different tools & interfaces. Additionally, code formatting/linting, and dependency management were also integrated using GitHub Actions built-in support, which helped in automating these repetitive tasks thereby helping the team focus more on writing quality code. These outcomes were observed from the three case studies conducted and mentioned in previous sections.

GitHub Actions provided highly optimized & scalable runners in form virtual machines & containers for executing workflows. They were efficient & responsive which allowed faster execution of CI/CD and automation runs when compared to self-hosted infrastructure. Additionally, the declarative YAML used to write workflows was much simpler and more readable. This made it easier to setup and maintain CI/CD pipelines, and thereby reducing the time spent on configuration and maintenance tasks. This led to about 10% less time spent on the CI/CD and automation after GitHub Actions migration.

After migration, we did analysis of the time spent with GitHub Actions when compared to Jenkins before migration. For every 100 hours spent with Jenkins, we were spending 65 hours with GitHub Actions. Thus we can conclude that we achieved about 35% efficiency after migration.

## 7 Best Practices for Migration of GitHub Actions

### 7.1 Incremental migration

Incremental migration involves the transitioning of the workflows progressively. Rule of thumb here is to start with a simpler and low-risk task. In this way, we minimize risks & disruptions by allowing exhaustive testing and configuration for each flow before moving to the next one. Additionally, it enables continuous learning and improvement, making the migration process more manageable and efficient.

### 7.2 Security Best Practices

Use GitHub Actions secrets management to store API keys, and passwords, ensuring they are not hardcoded in workflows. Monitor and assess audit logs and workflows logs to monitor for any unusual activities and any exposed secrets. Using the principle of least privilege, regularly review permissions for actions and workflows. Keep the actions up to date to avoid any vulnerabilities, and possibly use signed actions wherever possible. Another way of securing workflows is pin 3<sup>rd</sup> party actions. To ensure that the source code of the version you're using hasn't been altered you must pin the action to the full-length commit SHA. The value of commit SHA is unique for each commit and is immutable. This ensures that your GitHub Actions workflows will always use the same release of the Action.

## 8 Common Challenges and Solutions

### 8.1 Technical challenges

Migrating to GitHub Actions offers exciting opportunities to modernize automation and CI/CD pipelines, though it may pose a few technical challenges. Translating existing automation & CI/CD configurations to the new syntax does involve careful planning and validation. Given the goal is to maintain or enhance security posture while migrating, configuring secrets and permissions securely is essential. For someone not familiar with GitHub UI and system and/or declarative YAML syntax, it may be a bit of a learning curve. Thorough testing and documentation are necessary to ensure functionality and team readiness. Tackling these challenges systematically can lead to a successful migration of GitHub Actions.

### 8.2 Organizational Challenges

One of the organization challenges faced in managing change resistance by team or team members accustomed with existing tools. Additionally, coordinating across multiple teams and aligning on newer workflows and best practices can get complex. Allocate sufficient resources and time for smooth transition to avoid overburdening and maintaining productivity during migration. Organize Tech-talk(s), office hours for collaboration and knowledge sharing. Comprehensive training and upskilling across board is required to ease the migration process and avoid any disruptions or delays.

## 9 Conclusion

### 9.1 Cultural Contribution

Over a period, which started with the migration process to its completion, the change/shift in the culture of the team was starkly visible. From being multiple teams, and to catering multiple profiles [Dev vs QA vs DevOps Engineer], we now noticed a uniformity in approach towards pipelines. The synergy that now drives the overall process is geared towards keeping the pipeline a standard tool, which is the frontline candidate for shipping a product.

The team was missing a mindset which was sustainable to maintain a DevOps lifestyle, with the advent of GitHub Actions, we notice that the pipeline is now a first-class citizen, with SecOps as a front-end driver. The security enhancements that we were able to introduce and provide have been a boon for the team.

## 9.2 Positive Impacts

As mentioned before, the change in mindset of the team has brought in multiple positives, e.g. with the advent of GitHub Actions, the infrastructure management, and the constant maintenance and scalability were now a worry of the past. The extra time can be utilized in improving the product and implementing new ideas, the QA can focus on product automation and the DevOps can enable the team to sustain a quality pipeline, which is not compromised by vulnerable artifacts. The standardization is a boon in terms of Audits and Quality Checks which come as a part and parcel of Enterprise Software teams.

The Return of Investment is an often-discussed relevant topic, the total investment in terms of time and cost, and the total man-hours saved by implementing GitHub Actions is a net positive number, which for privacy reasons cannot be discussed, overall, the time saved is 35% more than when using the older technology stack.

## 9.3 Quality beyond Testing

The biggest takeaway from this exercise was the improvement seen in Automation. Since the platform was common and the exposure was greater it allowed the Automation code to be validated by Senior Developers, giving the benefit of added reviews. It also brought in a common platform to secure the libraries and ensure that the latest and often the greatest were parts of automation frameworks. This led to better quality and stability in framework and in turn the automation caught more bugs than before.

## 9.4 Future Trends

The integration with CoPilot, is something to which the team is looking forward to, the hope that AI will bring an efficiency jump and that the pipelines will be able to detect a CrowdStrike like failure well in advance.

# 10 References

Mayank Sharma and Indrajit Rajput, "Enhancing Continuous Delivery Using DevOps Practices," *Journal of Systems and Software* 136 (2018): 110-119.

Jenkins Documentation. "User Handbook Overview." Last modified 2024. Accessed July 14, 2024. <https://www.jenkins.io/doc/book/getting-started/>.

Jenkins Documentation. "Pipeline Documentation." Last modified 2024. Accessed July 14, 2024. <https://www.jenkins.io/doc/book/pipeline/>.

Thomas Dohmke, "100 Million Developers and Counting," The GitHub Blog, January 25, 2023, <https://github.blog/2023-01-25-100-million-developers-and-counting/>.

GitHub. "GitHub Actions: Overview." GitHub Docs. Accessed July 14, 2024. <https://docs.github.com/en/actions>.

Smith, John. "Automating Workflows with GitHub Actions." *DevOps Journal* 12, no. 3 (2020): 45-58.

Heller, Priscila. *Automating Workflows with GitHub Actions: Automate software development workflows and seamlessly deploy your applications using GitHub Actions*. Packt Publishing Ltd, 2021.

Rautiainen, Olli. "GitHub Enterprise and Migration of CI/CD Pipelines from Azure DevOps to GitHub." (2023).

# AI-Driven Techniques for Noise Filtration in Software Validation Logs

**Tarun Arora**

[tarun.arora@intel.com](mailto:tarun.arora@intel.com)

## Abstract

This paper presents a comprehensive examination of the enablement and integration of Natural Language Processing (NLP) and Random Forest Classifier (RFC) techniques to streamline log file diagnostics within a pivotal software development project. This novel integration has significantly improved the accuracy of classifying errors and informational queries in log files, marking a considerable advancement over traditional diagnostic methods. Our approach leverages advanced NLP to efficiently process and interpret the extensive, complex data within log files. In tandem with the robust classification capabilities of RFC, our method identifies and categorizes failure signatures with remarkable precision.

The effectiveness of our methodology and its potential to substantially reduce manual intervention in system diagnostics are underscored by these results. This innovation not only advances the field of software diagnostics into a new era characterized by automation and precision but also establishes a strong foundation for future technological progress. As technology evolves, the insights from this research have the potential to transform the maintenance and reliability of complex computing systems, indicating a significant paradigm shift in the practice of technological diagnostics.

## Biography

*Tarun Arora is working as senior engineer with Intel Technologies Limited, India. He is a seasoned change management and analytics professional with qualified rich experience working with global teams and fortune 500 companies across various domains. He led efforts to drive project management, quality management, customer delivery, business intelligence solutions and handcrafted process automation solutions. Successful projects with savings ranging in million dollars trusted the stakeholders in the technical skills and leadership qualities.*

*He is graduated in Industrial Production Engineering, with continued skills enhancement through various courses including Executive Program in Business Intelligence, Six sigma black belt, Master of Total Quality Management, Agile practitioner etc. supported the vision of delivering value through service.*

## 1 Introduction

Platform development projects are validated for thousands of use cases by various system integration and validation techniques. Product validation is a continuous phenomenon with multiple test cycles executed over a period till the time Platform Validation (PV) milestone is granted. Each of the tests executed as part of these validation cycles generates validation logs which vary in types and sizes. Test types could be basic acceptance test, sanity test, stability, functional, Tape In, pre-integration and many others. These tests generate various log files, including .log, .csv, .err and each log has varied format. Even .log files generated for different tests have different formats. More importantly, these test cycles generate hundreds of logs weekly. Hence this is the classic case of Big Data 3 V - volume, velocity, and variety. Refer Figure1 (3V of Big Data).

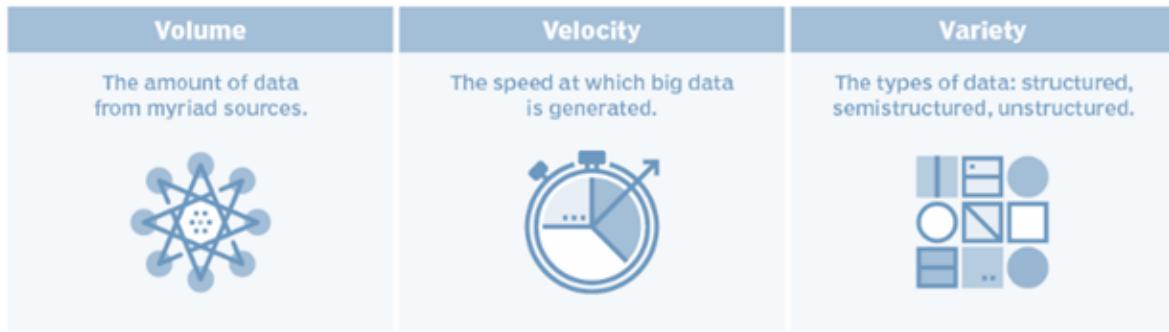


Figure1

## 2 Problem Description

Due to the 3V challenge of the validation logs data, the identification of real errors vs the noise elements in data is manual process and sometimes assisted with Regex operations. However, these Regex operations come with maintenance and become a nuisance for any new engineers in team. Also, there is always a possibility of missing errors due to human oversight. As reasoned above, overall efforts of filtering the noise from logs data and identifying real errors requires significant effort and time. This eventually leads to unproductive efforts and high triage or debug time which spans to ~1hrs to 2hrs per logs generated for a given test.

## 3 Solution

Quality and validation engineers worked together to develop a machine learning solution capable of predicting errors within validation logs and filtering out noise. The team identified that natural language processing (NLP) combined with classification modeling was the optimal approach. After creating and validating several models, the team successfully developed a proof of concept for the most effective machine learning model. This model automates the process of noise filtration and error triage in the logs. The image below illustrates two distinct workflows. The upper portion depicts the existing process, which involves a combination of manual effort and semi-automated tasks to identify error signatures. In contrast, the lower portion of the image represents the proposed integration of a machine learning model, designed to achieve a more efficient and effective end-to-end error detection system. Refer Figure2 below:

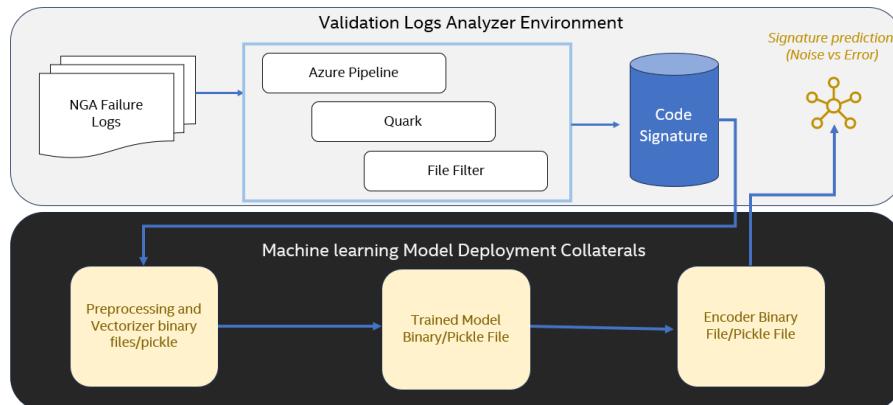


Figure2

## 4 Machine Learning: End to End Deployment Flow

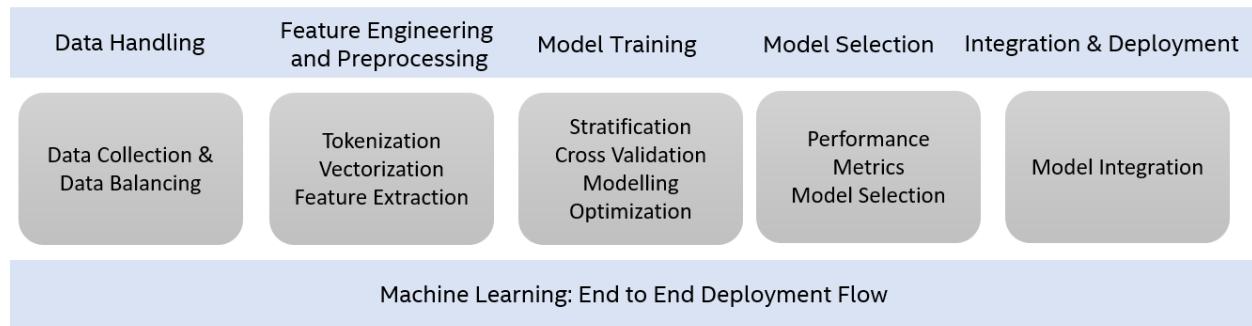


Figure3

## 5 Data Handling

The meticulous process of enhancing the triage tool for log file analysis started with a structured approach to data management. This process is pivotal for ensuring the foundation of our analysis is accurate, dependable, and tailored for optimizing diagnostics efficiency:

### 5.1 Data Collection

Logs dataset comprises essential elements such as log filenames, descriptors of queries, classifications of queries as errors or informational, alongside the substantive text of log signatures. "Table 1" provides a clear and organized overview of the dataset for easy reference and analysis.

File Name	Query Name	Query Type	Signature Text
XXX.log	ABC	error	XML oneof Check:

Table1

### 5.2 Guaranteeing Uniqueness of Data

To maintain the fidelity and integrity of our analysis, a crucial early step involves the removal of any replicative data entries. This ensures that each piece of data analyzed is unique, avoiding skewed results from duplicated information.

### 5.3 Streamlining Data Relevance:

The dataset undergoes a rigorous refinement process, where non-essential information is methodically stripped away. By isolating and removing columns that do not directly contribute to the failure diagnosis process, we focus our analysis on the most impactful and pertinent data elements.

### 5.4 Strategic Data Balancing:

Strategic Data Balancing plays a crucial role in addressing the inherent imbalance between 'error' and 'info' query types within our dataset.

**Before Balancing Data:**

INFO	ERROR
1488793	192565

To create a dataset conducive to unbiased machine learning model training, we implement a down sampling strategy. This involves equalizing the quantity of 'error' and 'info' categories by reducing the instances of the predominant class.

Mathematically, if the majority class has ( $N_p$ ) instances and the minority class has ( $N_l$ ) instances, we randomly select ( $N_l$ ) instances from the majority class to achieve a balanced dataset of ( $2N_l$ ) instances. This harmonized class representation is essential for preventing model bias and ensuring that our analysis is both accurate and reflective of true log file dynamics, augmenting the triage tool's diagnostic capabilities.

**After Balancing Data:**

INFO	ERROR
192565	192565

Also, the StratifiedKFold method is employed with  $n\_splits=5$ , ensuring that our data is divided into 5 distinct folds or splits for cross-validation, while maintaining an equal proportion of each class within every fold. StratifiedKFold, essential for handling imbalanced data, allows for a more reliable estimation of model performance by ensuring that each fold is a good representative of the overall class distribution. This technique is pivotal for reducing bias and variance in our model evaluation, supporting the authenticity of our performance metrics.

Stratified K-Fold cross-validation was used to validate the model's performance, defined by the formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where (TP), (TN), (FP), and (FN) represent true positive, true negative, false positive, and false negative predictions, respectively.

## 6 Feature Engineering and preprocessing

Given the nature of log files, typically unstructured and cluttered with irrelevant information, preprocessing is a crucial step. The goal here is to convert raw text into a numerical format that machine learning models can understand. To achieve this, we first employ Regular Expressions (Regex) to meticulously extract significant patterns and entries from the log files. Regex allows us to sift through large datasets and pinpoint relevant information by matching specific patterns.

### 6.1 Our preprocessing pipeline involves the following key steps:

#### 6.1.1 Tokenization and Stop Word Removal:

Tokenization involved breaking down the cleaned log entries into individual words or tokens. Following tokenization, we removed stop words - common words such as "the," "is," and "in," which typically do not contribute to the distinctiveness of log messages. This step was crucial for reducing the dimensionality of our data and improving the focus on keywords that could be indicative of system failures or errors. We utilize the `get_stop_words('english')` method to achieve this.

#### 6.1.2 Lowercasing:

Converts all text to lowercase to ensure consistency, as machine learning models are case sensitive.

### 6.1.3 Lemmatization:

Each token then underwent lemmatization, a process of reducing words to their base or dictionary form. For instance, "running" becomes "run". We employed the `getLemma` function for this purpose, focusing on nouns as primary entities in log data. This standardization of vocabulary significantly contributes to the efficiency of the subsequent feature extraction phase, which involved transforming our processed textual data into a numerical format.

For a word ( $w$ ) in our text ( $T$ ), if ( $w$ ) is not a stop word and its lemmatized form is ( $\text{lemma}(w)$ ), our filtered text ( $T'$ ) can be represented as:

$$T' = \{\text{lemma}(w) | w \in T \wedge w \notin \text{stopwords}\}$$

## 6.2 Feature Extraction:

The TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer was employed here, converting the corpus of preprocessed logs into a matrix of TF-IDF features. The TF-IDF metric represents the importance of a word within a document relative to a given corpus, balancing the frequency of the word in the document against its commonness across all documents.

Mathematically, TF-IDF for a term ( $t$ ) in a document ( $d$ ) within a document set ( $D$ ) is calculated as:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

where ( $\text{TF}(t, d)$ ) is the Term Frequency, representing the number of times term ( $t$ ) appears in document ( $d$ ), and ( $\text{IDF}(t, D)$ ) is the Inverse Document Frequency, with ( $N$ ) being the total number of documents, and ( $\text{df}(t)$ ) denoting the number of documents containing term ( $t$ ), computed as:

$$\text{IDF}(t, D) = \log \frac{N}{|d \in D : t \in d|}$$

with ( $N$ ) being the total number of documents in the corpus ( $D$ ) and ( $|d \in D : t \in d|$ ) representing the number of documents where the term ( $t$ ) appears.

The logarithmic scaling of ( $\text{IDF}$ ) diminishes the weight of terms that occur very frequently across the corpus, hence prioritizing unique terms in each document. Incorporating n-gram ranges (1,2) enables the vectorizer to not only consider individual terms (1-gram) but also pairs of consecutive terms (bi-grams), enriching the feature set and potentially capturing more contextual information useful for model training.

## 7 Model Training

### 7.1 Training and Testing by Data Splitting:

We precisely tailored our dataset for the Random Forest Classifier, implementing a stratified split to ensure a representative mix of 'error' and 'info' query types in both training (80%) and testing (20%) sets. This approach, facilitated by the `train_test_split` function with a set `random_state`, guards against bias and upholds the integrity of our log file classification model's evaluation, significantly contributing to the enhancement of the triage tool's diagnostic accuracy and overall reliability. This meticulous preparation underpins our dedication to refining automated diagnostics.

### 7.2 Model Training Enhanced with Machine Learning:

Following the data preparation phase, we focus on the objective optimization and validation of our model's performance. This stage is characterized by a well-structured approach that leverages the `RandomForestClassifier`, integrated with `GridSearchCV` for hyperparameter tuning, and `StratifiedKFold` for robust cross-validation. This methodology is aimed at refining our predictive

model for enhanced accuracy in distinguishing between 'error' and 'info' query types in log files. Here is a breakdown of this meticulous process:

### 7.2.1 Random Forest Classifier:

A RandomForestClassifier instance (RF\_GS\_CV) is initialized as our base model. RFC is an ensemble learning method for classification that operates by constructing many decision trees at training time and outputting the mode of the classes (classification) of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set, offering a more generalized model.

The decision tree, the building block of RFC, works by selecting the best feature among a random subset of features at each node to split the data. This randomness, coupled with bagging (Bootstrap Aggregating) - where multiple models are trained on different subsets of the training data and then averaged - ensures that the bias-variance trade-off is balanced. Mathematically, the Gini impurity, a measure used by the decision trees in the RFC to decide how to split the data, is given by:

$$G = 1 - \sum_{i=1}^n p_i^2$$

where ( $p_i$ ) is the frequency of class (i) in the dataset.

### 7.2.2 Hyperparameter Optimization with GridSearchCV:

To fine-tune our model, we employ GridSearchCV, applying it to the RandomForestClassifier with a set of predefined hyperparameters - 'n\_estimators' (the number of trees in the forest) and 'max\_depth' (the maximum depth of each tree). This exhaustive search over specified parameter values is conducted across k-fold cross-validation to ensure the model's generalizability. Mathematically, the optimization aims to maximize a performance metric (F), defined over the hyperparameter space (H), as:

$$\hat{h} = \arg \max_{h \in H} F(h)$$

Where ( $\hat{h}$ ) represents the optimal hyperparameter setting.

The performance of the model was evaluated based on accuracy, and the best parameters were chosen based on cross-validation scores, ensuring robust generalization to unseen data.

### 7.2.3 Model Training:

With the optimal hyperparameters determined, we train the RandomForestClassifier on the training subset transformed via TF-IDF vectorization, striving for a model that accurately discerns between 'error' and 'info' queries based on the textual content of log files.

## 8 Model Selection

The evaluation employed a suite of performance metrics—accuracy, precision, recall, and the F1-Score—each reflecting a distinct aspect of the model's classification capabilities:

- Accuracy: Serving as a primary measure, accuracy quantifies the overall correctness of the model's predictions across the test dataset, offering an aggregate perspective on its performance.
- Precision: This metric assesses the model's reliability in identifying true positive instances among all positive predictions, crucial for scenarios where the cost of false positives is high.

- Recall (Sensitivity): Recall measures the model's capacity to detect all actual positive instances, paramount in understanding its effectiveness in identifying relevant classifications without omission.
- F1-Score: The F1-Score amalgamates precision and recall into a single metric, providing a balanced view of the model's precision and recall capabilities by computing their harmonic mean.

We conducted a thorough evaluation of various diagnostic models, including logistic regression, naive Bayes classifier, support vector machine, random forest, and gradient boosting, and we also optimized their hyperparameters.

Based on the performance metrics, we narrowed our final selection to two candidates: Random Forest and Support Vector Machine.

Finally, we compared the performance of RandomForestClassifier (RFC) with that of Support Vector Machine (SVM) to identify the most effective tool for classifying 'error' and 'info' query types in log files. The analysis incorporated a detailed examination of predictive accuracy, particularly focusing on the incidence of false positives and false negatives, alongside metrics of precision and recall, to offer a granular understanding of each model's efficacy.

- SVM Performance Analysis:
  - The SVM model yielded a false positive and negative rate of approximately 0.5%, indicating a scenario where 0.5% of cases might be misclassified as 'info' when they are 'error', and vice versa. This rate, while low, underscores a critical challenge in distinguishing between the two query types with high reliability using SVM.
- RFC Performance Insights:
  - In contrast, the RFC model exhibited a significantly lower false-negative rate for 'error' predictions at 0.05%, albeit with a false positive rate for 'info' at around 3%. This suggests that while RFC is markedly adept at identifying 'error' queries with high coverage, it also presents a tendency to misclassify some 'info' queries as 'errors'.
- Precision and Recall Metrics:
  - Precision, evaluating the correctness of predictions for a given label, and recall, assessing the model's proficiency in accurately identifying correct labels out of the total population in the given data, served as pivotal metrics. The RFC model showed commendable precision in its predictions, coupled with a superior recall rate, highlighting its ability to effectively identify 'error' classifications from the dataset.

Following this comparative analysis, Figure3, highlights the performance metrics of various models evaluated during the process with focus on SVM and RFC models, providing a visual summary of their predictive efficacy, error handling, and overall capacity to accurately classify log file queries.

Metrics	TFIDF_SVM	TFIDF_Logit	TFIDF_NB	TFIDF_RS_GS_CV Est: 50 , Depth: None	TFIDF_GB Est: 100 , Depth: 10, learningrate: 0.5	TFIDF_SVM	TFIDF_RS_GS_CV Est: 50 , Depth: None
<b>FN</b>	<b>368</b>	<b>399</b>	<b>816</b>	<b>363</b>	<b>368</b>	346	169
<b>TP</b>	<b>60988</b>	<b>60949</b>	<b>59318</b>	<b>60997</b>	<b>60991</b>	60993	263069
<b>FP</b>	<b>263</b>	<b>302</b>	<b>1933</b>	<b>254</b>	<b>269</b>	258	1484
<b>TN</b>	<b>60660</b>	<b>60629</b>	<b>60412</b>	<b>60865</b>	<b>60857</b>	60882	36946
<b>FPR</b>	<b>0.60%</b>	<b>0.65%</b>	<b>1.36%</b>	<b>0.59%</b>	<b>0.60%</b>	0.56%	3.86%
<b>TPR</b>	<b>99.40%</b>	<b>99.35%</b>	<b>98.84%</b>	<b>99.41%</b>	<b>99.40%</b>	99.44%	99.94%
<b>TNR</b>	<b>99.57%</b>	<b>99.51%</b>	<b>96.90%</b>	<b>99.58%</b>	<b>99.57%</b>	99.58%	96.14%
<b>FNR</b>	<b>0.43%</b>	<b>0.49%</b>	<b>3.10%</b>	<b>0.42%</b>	<b>0.43%</b>	0.42%	0.064%

Table2

To complement these quantitative assessments, a Confusion Matrix was constructed, offering a visually detailed account of the model's predictive outcomes. The confusion matrix delineates the distribution of true positives, true negatives, false positives, and false negatives, providing nuanced insight into the model's classification behavior. This visualization aids in identifying patterns or biases in the predictions, further contextualizing the numerical metrics. Following this, Figures 4 and 5 are presented, highlighting the confusion matrices for RFC and SVM models, respectively. These matrices serve as crucial tools for visually interpreting the performance of each model in classifying 'error' and 'info' queries in log files, allowing for a deeper understanding of their predictive accuracy and areas for improvement.

	<b>error</b>	<b>info</b>
<b>error</b>	<b>36946</b>	<b>169</b>
<b>info</b>	<b>1484</b>	<b>263069</b>

Table3: Random Forest Confusion Matrix with Stratification Strategy

	<b>error</b>	<b>info</b>
<b>error</b>	<b>60882</b>	<b>346</b>
<b>info</b>	<b>258</b>	<b>60933</b>

Table4: SVM Confusion Matrix with Down sampling Strategy

This array of performance metrics, coupled with the confusion matrix, furnishes a comprehensive understanding of the RFC model's predictive prowess.

The comparative analysis of RFC with traditional Support Vector Machine (SVM) models demonstrated the former's superior performance in minimizing false-negative rates and enhancing precision in error query identification. This underscores our methodology's effectiveness and its potential to significantly reduce manual intervention in system diagnostics.

## 9 Integration and Deployment

The integration phase fuses the trained RandomForestClassifier (RFC) model with Triage tool, granting it the capability to autonomously identify and categorize failure signatures in log files through advanced NLP techniques and RFC's classification power. This harmonization significantly boosts Triage tool's automated log analysis efficiency, aligning with project goals to elevate system diagnostics' precision and reliability.

- The selection and implementation of the RFC-based model in Triage tool highlighted substantial improvements in the tool's performance. One of the major outcomes was the ability to create more accurate and granular failure signatures without explicit pre-programming. This advancement meant a decrease in the need for manual interventions during the failure analysis phase. Additionally, the RFC model improved the predictive accuracy of Triage tool, allowing it to adapt to new and unseen failure modes.
- Implemented Standalone Model and infra for testing Model in Millions of files from every project.
- Analyzed log in Host Patch Update logs and identified many occurrences of Patch upload failure.

By presenting these results, we contribute novel insights into automatic log file analysis, underscoring the viability of employing advanced machine learning algorithms and NLP techniques for refined diagnostic endeavors. The clarity and precision of these results underscore our commitment to advancing the field and offer a significant contribution to the technological community's understanding of automated system diagnostics.

After rigorous training and meticulous hyperparameter tuning, the embedded Random Forest Classifier (RFC) model has enhanced Triage tool's ability to process raw textual data. It now efficiently extracts actionable insights, driving maintenance strategies to new levels of effectiveness.

This pivotal step underscores our commitment to leveraging innovative technology for enhancing software diagnostics and ensures a broad impact on system reliability and maintenance methodologies across the technological landscape. The figure below illustrates the integration of the machine learning model with the validation log analyzer within the final environment.

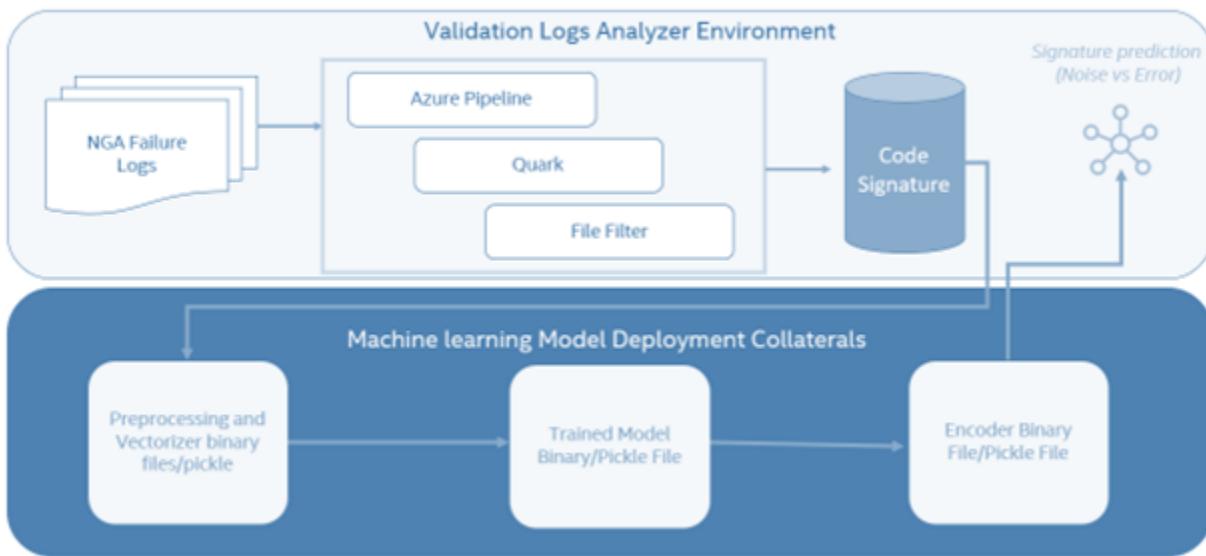


Figure4

## 10 Conclusion

We embarked on a comprehensive assessment of integrating Natural Language Processing (NLP) and RandomForestClassifier (RFC) techniques for streamlining log file diagnostics. This integration, the first of its kind, notably enhanced the accuracy of classifying errors and information queries in log files, signifying a substantial step forward from traditional diagnostics methods. Our approach utilized advanced NLP to efficiently process and interpret the vast and complex data contained within log files, coupled with the classification power of RFC, to identify and categorize failure signatures with unprecedented accuracy and efficiency.

As outlined in the problem description section, the analysis of each log file generated by a given test typically requires 1 to 2 person-hours, depending on the size of the log. With an average of 10,000 log files being generated weekly, we can conservatively estimate a potential time saving of up to 10,000 to 20,000 person-hours per week. This estimate assumes that the implementation of our solution would eliminate the need for manual analysis, thereby saving a significant amount of effort.

In conclusion, this machine learning project asserts the significant potential of marrying NLP with RFC within the domain of log file analysis. It not only propels the field of software diagnostics into a new era marked by automation and precision but also sets a robust foundation for future advancements. As technology continues to advance, the insights from this study bear the potential to revolutionize the maintenance and reliability of complex computing systems, signaling a radical shift in how technological diagnostics are approached and implemented.

# Shifting Other Than Left or Right

Jon Bach

*jbtestpilot@gmail.com*

## Abstract

You've likely heard of "Shift-Left" -- the DevOps principle of moving testing closer to Development with programmatic approaches like Build/Code/Plan/Test pipelines.

The complementary DevOps dimension is "Shift-Right" -- moving testing closer to Production with programmatic approaches like Release/Deploy/Operate/Monitor.

In my work as a Program Manager who works with Developers, Testers, Business Intelligence, Legal, Operations, Compliance, Localization, and every adjacency you can think of, I've noticed a third and fourth dimension we don't talk enough about -- Shifting IN to the details and OUT to the big picture -- getting perspective so you know how best to marshal your Left and Right resources.

## Biography

Jon's been in Tech since 1995, starting in Customer Support for a commercial real estate dial-up service. He's currently a Senior Program Manager Quality Manager and consultant for a small software development studio called ProphetTown. His longest role was at eBay where he worked for 13 years a Quality Director and Program Manager. He's also been a full-time employee at Microsoft during Y2K, and served on contracts at HP, Adobe, AT&T, WebMD, Getty Images, Alaska Airlines, McGraw-Hill, and more. He's the co-inventor (with his brother James) of Session-Based Test Management – a way to manage and measure Rapid Software Testing. Jon loves turning rhetoric and philosophy about software development into action (baking half-baked ideas), and frequently posts on LinkedIn about software quality and program management.

*Copyright Jon Bach, August 5, 2024*

## Introduction

I frame this paper with a model of thinking about how to examine aspects of a software risks and problems from what I learned in over 13 years at eBay as a Quality Program Manager and customer advocate

If shifting left is being able to find problems earlier, and shifting right is about learning from Production, it's time there was a paper about how we can see not just the Left or the Right, but zoom In and OUT to see the significance of one tree or the beauty of the WHOLE forest.

## The Problem

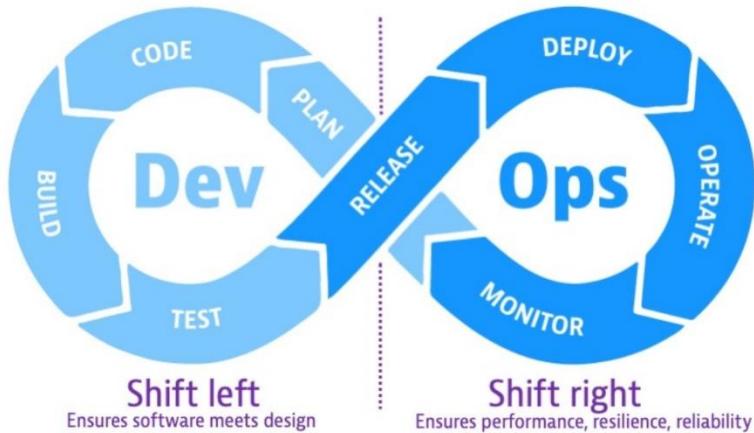


Diagram courtesy [Dynatrace website](#) (article by Saif Gunja)

This is a classic representation that's a model of Development and Operations in terms of moving testing closer to Development with programmatic approaches

This seems like Waterfall Development with a twist – literally. This diagram keeps coming back to the MIDDLE – both shifts might be good, but their captions might benefit from some context and perspective.

“Shift Left” might really mean: “We do our best to get it right before it goes out”

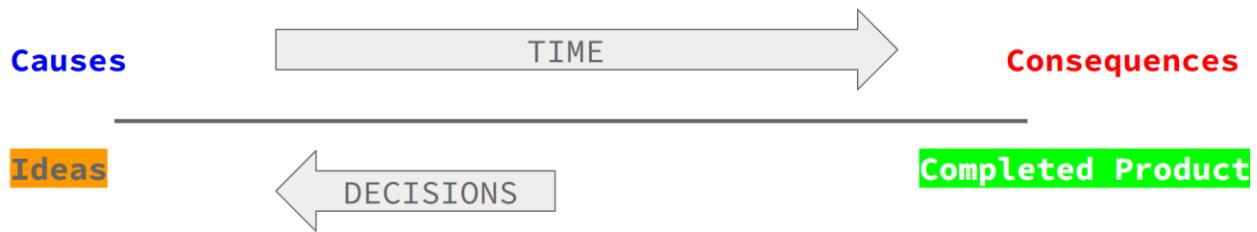
“Shift Right” might really mean: “We do our best to learn from what happens in the field so that we can do better in the next iteration.”



LEFT is the DevOps principle of moving testing closer to Development with programmatic approaches like Build/Code/Plan/Test pipelines.

It's meant to be linear, with time going from left to right, from CAUSES to CONSEQUENCES, but there's also a spectrum between IDEATION and a COMPETED PRODUCT, from which things like decisions are meant to be moved closer to IDEATION to prevent the risk of costly rearchitecture and redesign.

The ultimate LEFT is looking at every line of code as it's built and seeing quick ways to prevent problems before they happen. This could mean unit tests and code reviews, advanced IDE syntax or pattern-checking features, continuous integration on every pull request, checkin, and merge, or recent claims that AI can find, assess, suggest, and create fixes to problems that haven't happened yet.



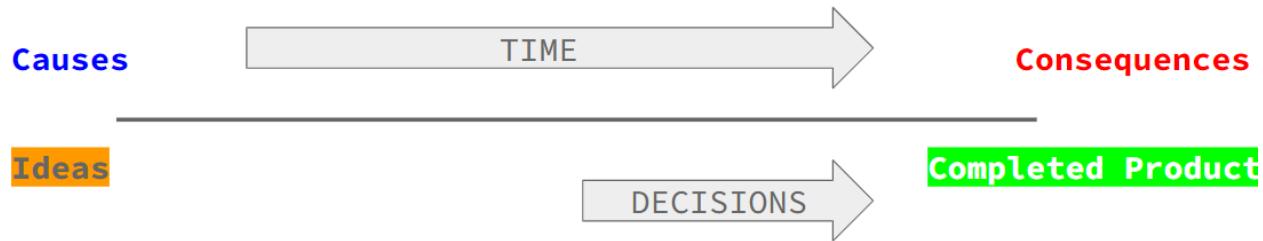
Now, let's take its complement – shifting RIGHT – the DevOps principle of moving testing closer to Production with programmatic approaches like Release/Deploy/Operate/Monitor.



Diagram courtesy [Dynatrace website](#) (article by Saif Gurja)

It's also meant to be linear, with time still going from left to right, from CAUSES to CONSEQUENCES, also against the backdrop of IDEATION and a COMPETED PRODUCT, but where decisions are meant to be moved AWAY from Ideation because there's not enough data or context to make a call.

The ultimate shift RIGHT is Procrastination. Procrastination isn't bad. Delaying a decision or an action gives people a chance to learn, acquire better equipment and facilities, better tech, and less \*guessing\*. Working on a Minimum Viable Product or a 1.0 where you expect early adopters from your prototyping and incremental development is really shift RIGHT. You see how people like it first. Sure, if you could remove the cause of a bad event, that would be great, but there's a paradox: we want to go as far back as we can to make good things happen, but we don't want to make any decisions until we know how it all turns out! If you don't have a time machine, you will have to ITERATE to adjust your INVESTMENT.



## Alaska Airlines Flight 1282 (January 5, 2024)

Let's anchor these concepts in a recent example – Alaska Airlines Flight 1282.

There's an excellent report up on the National Transportation Safety Board website (<https://www.ntsb.gov/investigations/Pages/DCA24MA063.aspx>)

*"On January 5, 2024, about 1714 Pacific standard time, Alaska Airlines flight 1282, a Boeing 737-9, N704AL, returned to Portland International Airport (PDX), Portland, Oregon, after the left mid exit door (MED) plug departed the airplane leading to a rapid decompression. The airplane landed on runway 28L at PDX without further incident, and all occupants (2 flight crewmembers, 4 cabin crewmembers, and 171 passengers) deplaned at the gate. Seven passengers and one flight attendant received minor injuries."*

The report has the following elements:

- Crew Experience and History of Flight
- Recorders: Cockpit Voice Recorder and Flight Data Recorder
- Operator and Airplane Information
- Mid Exit Door Plug Description
- Cabin Pressurization/Cabin Description
- Materials and Structure Examination
- Manufacturing Records/Human Performance
- Safety Actions

### Working on the LEFT (*pushing decisions closer to ideas and causes*)

A lot of this is evidence of working toward the LEFT. Considerations like the relationship to the assembly line, time pressure, psychological safety, having information to do the right thing; knowing there's a problem and being able to solve it and WANTING to solve it.

### Working on the RIGHT (*learning from the consequences of incidents in the completed product*)

In other words, are we as software professionals on the lookout for signs and symptoms of problems in the field? (For example, the NTSB looked at maintenance logs indicating a pressure controller light had illuminated on three previous flights.)

In software development, shifting right could mean going from creating FUNCTIONS to verifying SOLUTIONS. You can't have functionality without thinking "Does anyone NEED this?"

## Functionality

---

## Solutions



### IN vs OUT

Now let's look at two other important dimensions that few people talk about as a representation of something other than LEFT or RIGHT.

In versus Out is akin to using a microscope or a telescope.

One context-centering question we could ask is:

- Who's looking? What problem are they trying to solve?

For the NTSB investigator, it's these:

- What kinds of things have to go into my report?
- What are the politics and legal implications?
- What are the forces that are guiding the report?
- What's relevant and what's irrelevant?

**Shifting IN (using a microscope) is akin to the following:**

- Focusing on the fine details
- Making a "Situation Room"
- Writing the "play-by-play" on how it happened
- Seeing what the "bolts" (literally, in this case) look like

It's also taking time to understand how the work happened:

- How do people read checklists?
- What are (or were) they told?
- Are they given too much work?
- What are they told vs. what do they actually do?
- Understanding the staff's daily experience

**Shifting OUT (using a telescope) is akin to the following:**

- Addressing general policy changes that affect people's behaviors,

You can go out in different directions, for example -- toward Strategy or

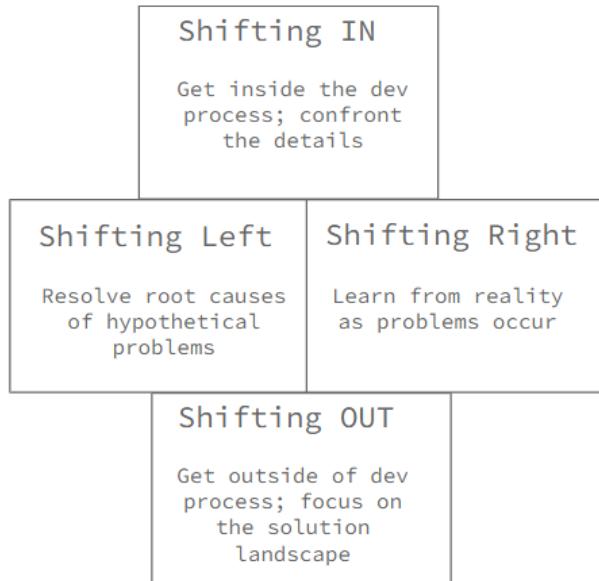
toward Big Picture, which includes timelines, system components, etc.

Shifting out could also be about delegation or outsourcing -- letting go of the problem so you specifically and purposefully become an outsider to get critical distance. You might deliberately find value in shifting your involvement.

## Applications of these dimensions

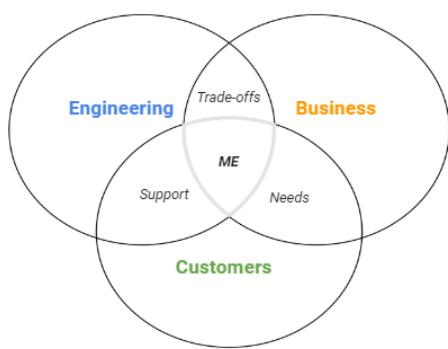
Shifting In or OUT is a complementary dimension to LEFT and RIGHT.

You can look at the dimensional focus like the movements of a joystick.



When I shifted careers from a QA Manager to a Program Manager after about 20 years, my perspective went from paying attention to one project to many. I went from finding BUGS in a PRODUCT to finding RISKS in multiple PROJECTS that were needed to comprise an enterprise PROGRAM. A program is a collection of deliverables from different teams who need a coordinator of their individual solutions to make one big solution -- much like individual features comprise a whole product.

Let's look at three major overlapping domains in Software Development:



When I was at eBay working on the Customer Survey platform, I needed to deal with many teams to either get the big picture of patterns of failure that were reported, or drill into specific bugs customers were reporting. I shifted Left, RIGHT, IN and OUT and it occurred to me I was in a new role that I'd never heard of in 30 years of software development, so I came up with a name for it:

## Customer Advocacy and Response Engineering

What can we learn from customers and what are we doing about it?

What I think I have here in this set of perspectives is a ***critical thinking tool on how to approach a project.***

**Between Engineering and Business**, I worked to consider and coordinate **Tradeoffs** – for example, Engineering can only do X, even though the business wants X, Y, and Z.

**Between Business and Customers**, I worked to collect and assess **Needs** – for example, Business needed more items on the site, and Customers wanted an easier time to list them.

**Between Customers and Engineering**, I worked to see what **Support** needed to look like – for example, Engineering needed more specificity about Production problems to get to the root cause more quickly, which worked well to support customers in rapid and clearly communicated bug fixes.

If we anchor these three sets of relationship back to something like an NTSB report, we see all 4 ways of coping with a problem:

- 1) IN: You get all the details of the problem and steep yourself in it. You become the expert, the ultimate insider, to get really close to it.
- 2) OUT: Think about its context and the dynamics that underlie it. Think about large flows of energy and flows through the system, as well as the players involved.
- 3) LEFT: Look at it in terms of its beginnings. Where did this problem come from? What was it like right when the prob started?
- 4) RIGHT: What did the problem eventually become? What did it manifest as?

At any time on any project we have to choose a frame of reference. We need to decide what's the object we're studying? For example, there's a perspective that a person in charge of QA at Boeing has AFTER the January event happened. They know what they know from this report and then they ask "Now what?" Is that person trying to go left / right / out / in, or is it the NTSB the people who are trying to write a report and it's the person in CHARGE of that team who needs to go left / right / out / in.

When I was a Director of Program Management at eBay on the customer survey platform team, my point-of-view was usually like the senior leaders at Boeing who read the NTSB report

But even then, there are opportunities to pivot – or, like a camera, to pan or zoom.

- 1) Pan LEFT -- how was this pattern of bugs created? Was there a recent release which had an infrastructure hiccup during the rollout? What is the relationship of a team rolling to Production and the Operations personnel? Was there time pressure? When someone did the wrong thing to cause a rollout problem, did they not know the process, or did they know but were overwhelmed.
- 2) Pan RIGHT -- if aspects of the release were screwed up, how could we know? Are we on the lookout (monitoring and alerting, for example) for signs and symptoms of stuff going wrong in the field? Do we hear about every problem, or do we fix things in silos one at a time and don't see patterns, perhaps even in other similar releases that day?

- 3) Zoom IN – it's all about the details in exactly how things happened. It's the play-by-play, the timelines of events. It's about seeing the code or the ramp parameters, the server stats, understanding procedure and how people read and follow it. It's about what they are told or the fact that they might have been given too much work; what they were told to do vs. what they actually did. It's about understanding the daily experiences of the people involved.
- 4) Zoom OUT – this could be about corporate policy or changes that affect people's behaviors. You can go out in different directions -- toward strategy or the big picture; toward large-scale timelines, or the components of systems and how they interoperate. Shifting OUT is also about the meat-cognitive plan of what kinds of things have to go into an incident report. Maybe there are politics or legal implications. What are the forces that are guiding the report? What's irrelevant and relevant? What absolutely HAS to go not the report vs. what would I like to go in the report? Maybe it even involves coordinating or considering a whole panel of experts to qualify the report.

## Summary

This year's PNSQC theme ("The Future is Now"), could mean that this Customer Advocacy and Response Engineering role I served at eBay is the future of how Quality gets assessed – an appropriate combination and teamwork of Testing perspective and Program Management perspective, aided by not only appropriate shifts either to the LEFT or the RIGHT, but knowing also when to use a telescope or microscope to shift IN or OUT.

## References

Tech Beacon. "Why software development needs shift not left or right" February 2019. Michael Giacometti. <https://techbeacon.com/app-dev-testing/why-software-development-needs-shift-not-left-or-right>

National Transportation Safety Board. "Alaska Airlines Flight 1282 Investigative Hearing" <https://www.ntsb.gov/news/events/Pages/Alaska-Airlines-1282-Inv-Hearing.aspx>

National Transportation Safety Board. "In-Flight Mid Exit Door Plug Separation" <https://www.ntsb.gov/investigations/Pages/DCA24MA063.aspx>

Dynatrace. "Shift left vs shift right: A DevOps mystery solved." May 2024. Saif Gunja <https://www.dynatrace.com/news/blog/what-is-shift-left-and-what-is-shift-right/>

Dynatrace. "What is DevOps? Unpacking the purpose and importance of an IT cultural revolution" February 2023. Saif Gunja <https://www.dynatrace.com/news/blog/what-is-devops/>

Also credit to James Bach for deep conversations and for helping me develop out the parameters of the idea. (Satisfice, Inc.: April 2024)

# Artificial Intelligence & Machine Learning in Test Data Management

**Praveen Bagare**

bagare@gmail.com

## Abstract

Artificial Intelligence (AI) and Machine learning (ML) are finding their roots in every stream of software development, be it self-driven cars, robo calls, meta humans or even digital clones. Test Data Management is no exception, significant use of ML for data discovery and profiling helps automate time consuming manual tasks into short, repeatable and effective outcomes. AI being leveraged to create Synthetic test data is just one example of how significant efficiencies can be brought into Test Data Management (TDM).

This paper will take you through key approaches to TDM solutions and how AI and ML can be applied across different stages of the Test Data Life Cycle (TDLC).

## Biography

*Praveen is a seasoned professional with over 20 years of IT experience in Program Management, Quality Assurance, and specialization in Test Data Management. He is passionate about solving complex challenges, driving innovation and excellence.*

*Currently, he is leading EPAM's Test Data Management Competency Center in the North America region and has successfully architected, implemented and Managed Testing and Test Data Management solutions across Fortune 100 companies in the USA.*

*He has authored a white paper on Test Data Management in Software Testing Life Cycle, is certified in multiple TDM tools, is ISTQB certified and is a senior member of IEEE.*

## 1 Introduction

Machine learning has the ability to understand and learn from large volumes of data, and AI has been evolving to quickly bring back relevant information to the user faster and with more accuracy than ever before.

As AI is making leaps not only in autonomous cars, customer center calls, automated kiosks, and many other areas, it is also advancing in the test data management space. The ability to analyze data, learn, and re-create new sets is not just a concept anymore, but a reality today.

Let's take a deep dive into what AI and ML have to do with test data management.

## 2 What is Test Data Management?

TDM is the process of provisioning data for testing ensuring it is of high quality, provisioned in suitable quantity, right format, appropriate environment, and in the stipulated time. [1] It typically offers services such as subsetting, masking, reservation, synthetic generation, copying, cloning, reservation, ephemeral solutions and so on. Test Data Life Cycle (TDLC) is the process originating from a Test Data request to the stage where it is provisioned to the user. Under the hood the requests may be provisioned through one of the service techniques mentioned earlier.

## 3 What is Artificial Intelligence?

Artificial Intelligence (AI), was a term coined by Stanford Professor John McCarthy in 1955. It was defined by him as “the science and engineering of making intelligent machines”. [2] In other words AI is the technology that enables machines to perform tasks that typically require human intelligence.

## 4 What is Machine Learning?

Machine Learning (ML), is the process where systems learn from the data and improve through a feedback loop thus improving accuracy in every iteration. Typically, the larger the sets of source data, the more accurate the outputs are.

## 5 AI and ML in TDM Solutions

TDM solutions can be categorized into three broad approaches. Each of them have their own pros and cons and can be influenced by AI and ML.

### 5.1 Approach 1 - Conventional, leveraging Masking

These types of solutions focus on how production data can be masked and moved safely into lower environments for testing, thus helping with compliance to regulations like HIPAA (Health Insurance Portability and Accountability Act) [3], General Data Protection Regulation (GDPR) [4] and others. The process is not straightforward and can involve many complex steps:

- Determine if there is an appetite for a masking tool or will the process have to be done manually
- Identify and onboard the TDM tool that fits the masking needs of the organization (some factors to compare include databases supported, on-prem/cloud accessibility, security, support offerings and others)
- Find the personally identifiable information (PII), Personal Health Information (PHI), Payment Card Information (PCI) and other sensitive elements in the databases, files and other systems.

- Get this list of identified elements reviewed and approved by the application teams, security and other key stakeholders.
- The next step is to identify algorithms that will be used to mask these elements. The algorithms need to be signed off by security and compliance teams as well, to avoid reverse traceability.
- Post that, these masking algorithms need to be run on a copy of production data, probably in a staging environment or gold copy environment.
- After that we get into validating the masking by comparing the masked PII elements with production data. Once done, the data set is either subset or fully loaded into the target testing region and handed over to the application team for testing.
- A feedback loop is typically introduced where any slipped sensitive data can be identified, fixed and updated right back into the original masking rules.

As you can imagine, there are a lot of pros and cons to the solution as it can be time consuming and expensive to execute but can bring in the assurance of safety. The magic quadrant TDM tools in the market can cost an enterprise millions of dollars to procure and effectively implement these solutions.

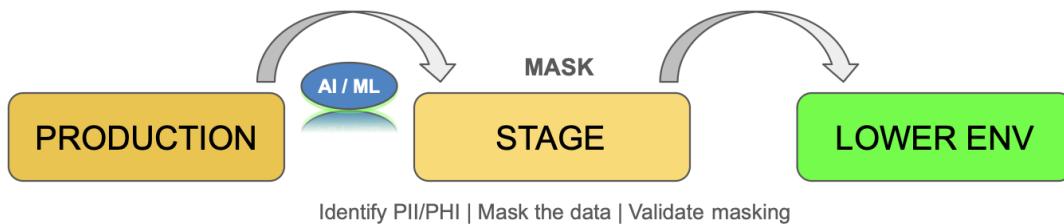


Figure 1 - Test Data Management leveraging masking

As seen in the image above AI/ML can play a part in identifying the PII/PHI elements and assigning appropriate generators/functions/algorithms to them. This speeds up the process and improves the efficiency significantly.

## 5.2 Approach 2 - New, leveraging synthetic data from scratch

This solution is relatively new and leverages the ability to examine the production database structure or the requirements and build synthetic data from scratch. These solutions are preferred when and where there are significant security concerns, such as in the healthcare and banking industries. Managing sensitive data becomes much easier because production data is not brought down to the lower environments. This may be a huge benefit when it comes to security and compliance team sign off. Typically, these solutions start by:

- Identify and onboard the right synthetic data generation TDM tool or tools.
- Understanding the table relationships in and across databases.
- Then adding further business rules into these functions or generators for data integrity.
- Assigning generators or functions to each column of data to be generated.
- Followed by generating sample test data sets and loading them to test environments.
- Validation of the data set through the application under test.
- Feedback loop to update the generators, functions and rules to create accurate data for testing.

The rules have to be well-defined, reviewed, approved, and tested. Commercially available tools can not only run these solutions on-prem but also in cloud or any other infrastructure. These tools allow the ability to write to the database directly or to files in different formats such as JSON, XML, and CSV, efficiently handling large volumes of data, potentially even millions of rows in seconds.

This solution is very effective in helping train machine learning models where the data is easily accessible to the learning system while addressing security concerns. Synthetic data may be a boon, but let's explore the other approaches too.

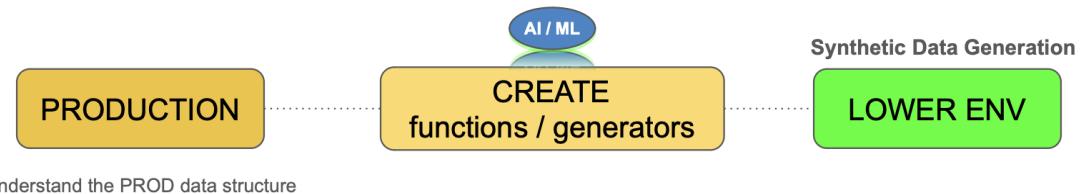


Figure 2 - Test Data Management leveraging synthetic data generation

As depicted in the above image AI and ML can play a key part in identifying and assigning the right functions or generators. This reduces human intervention, increases accuracy and speeds up the entire process.

### 5.3 Approach 3 - Futuristic, leveraging ML and AI to generate data

This approach examines production data and learns from it. The process can identify data types, the ratio of data distribution, and other mathematical entities. The machine learning process goes through the data and creates a model that can be used to regenerate similar data. Of course, there can be multiple iterations of learning, called epochs, which improve the model with every iteration. Once the model is available, security measures such as differential privacy can be applied. Once this model is secure, it can be moved to any lower environment and used to generate new sets of data. Here are the typical steps.

- Identify the subset of data on which you want to run the machine learning (biased, corrupt and irrelevant data rows can be eliminated for better performance).
- Configure the number of epochs (passes of training dataset through the algorithm), the relationships, pass through columns (e.g. policy number), custom requests (e.g. address) etc.
- Run machine learning (train) on the identified data set. This step can be Graphics Processing Unit (GPU) or Central Processing Unit (CPU) intensive.
- Move the differential privacy applied model to a lower environment where the data needs to be generated.
- Finally run the synthetic data generation (infer) to create desired volumes of data with similar attributes and distribution as the source data.

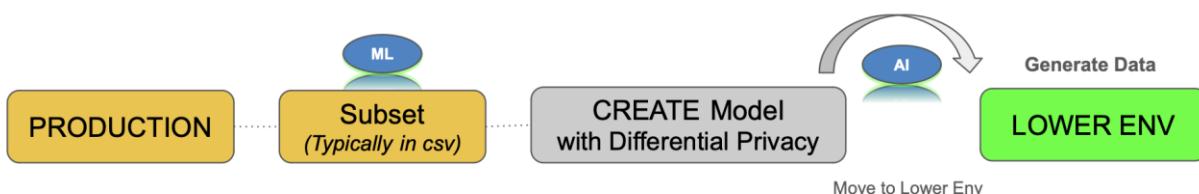


Figure 3 - Test Data Management leveraging machine learning and AI based generation

This solution comes with many benefits, including that the system automatically understands the source data through ML, which reduces human effort significantly. The generated data in the lower environment using AI has similar distribution as source, thus can be used for data analytics and business decision making. The data is not connected to production data, thus making it easier for Security and Compliance teams to sign off on these approaches.

However, there are some concerns, the generated data may not look very similar to the source data if the number of epochs are low. The process may have some hallucinations, and the accuracy may not be at 100%.

#### 5.4 Comparison

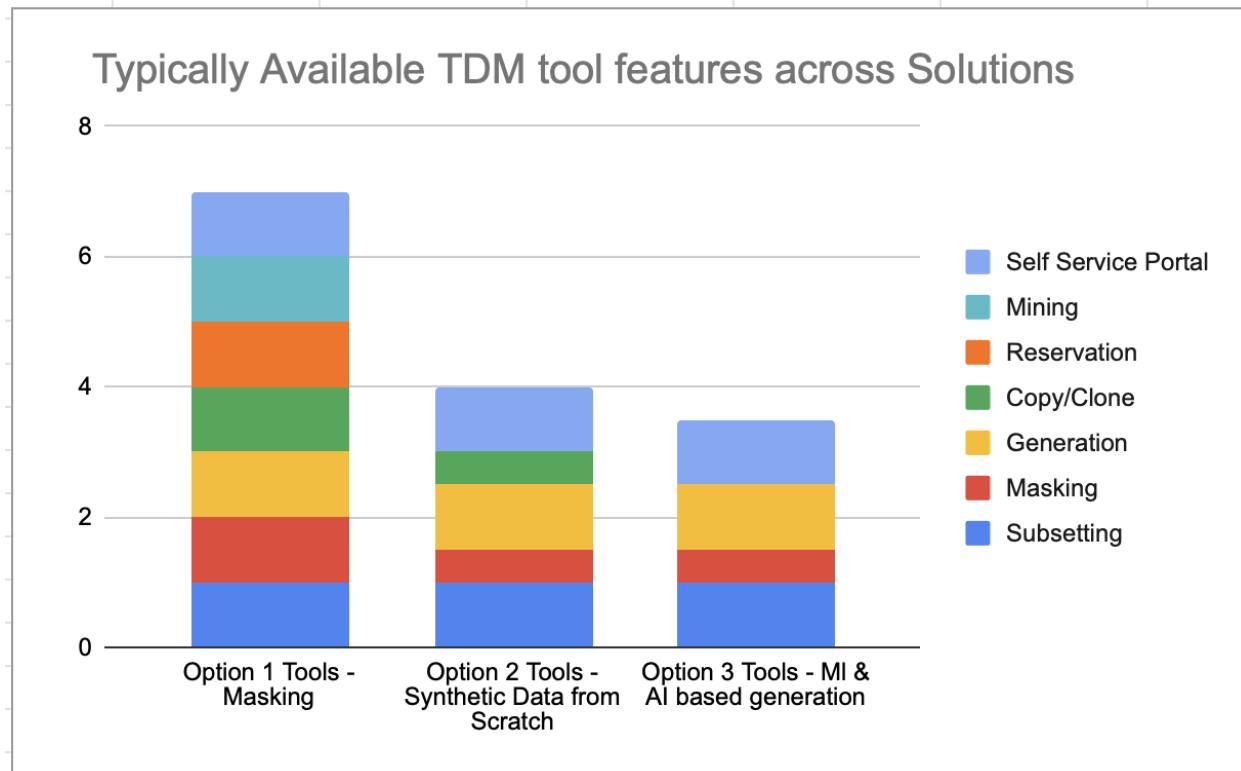
Let us compare the three TDM approaches side by side. Text highlighted in green are positive indicators.

	<b>Approach 1 (Masking)</b>	<b>Approach 2 (Synthetic data from Scratch)</b>	<b>Approach 3 (Synthetic data from sample data)</b>
<b>Cost</b>	High	Low	Medium
<b>Implementation complexity</b>	High	High	Medium
<b>Maintenance</b>	High	Medium	Low
<b>Security/Privacy</b>	Low	High	Medium
<b>Ensure Data Integrity</b>	Medium	High	Low
<b>Return on Investment</b>	Low	High	Medium
<b>Effort</b>	High	High	Medium
<b>Time</b>	High	Medium	Low

Table 1 - comparison of the three TDM approaches

## 6 TDM Tools

A wide variety of TDM tools are commercially available in the market for the approaches discussed above. Below are indicators as to how the different TDM services span across the tools.



Graph 1 - A graphical representation of TDM tool vs typically available features

A key point to note is that AI and ML tools may seem to have a limited coverage of services, but may be sufficiently able to solve the key asks from the client.

## 7 Case Study

Application under test: A simple two page web bank application for a user to submit an account opening form and be navigated to a confirmation page with a \$200 account opening bonus.

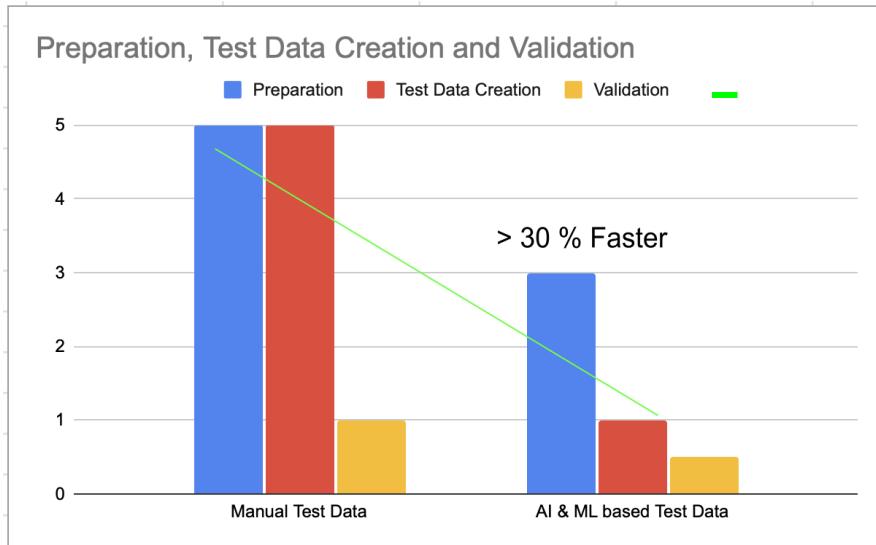
Database: A SQLite database with two simple tables, 'Application' and 'AccountBalance'.

What was done? In this case study Approach 3 was used (Synthetic Data generation using an AI and ML based TDM tool) to generate the Test Data for testing. The ML solution was used to train a model on the two tables with a few thousand rows of data. The model was then used to infer/generate the test data with similar ratios as source, in the testing database. If test data had to be provisioned manually, it would have been extracted, masked and loaded into the target environment. It was observed that the manual test data preparation effort was over 30% more as compared to the AI/ML solution for testing this simple application.

This example gives you an indicator to the amount of effort and cost savings that effective TDM can bring to the table & further get enhanced with AI and ML solutions in TDM.

Benefits:

- Faster, thus saving time, effort and money.
- No Personally Identifiable Information (PII) or Personal Health Information (PHI) or Payment Card Information or other sensitive information flows down to a non-prod environment, thus reducing the risk of data theft.
- The source data distribution is maintained in the generated data, thus it can be used for data analytics and business decisions.
- In fact this type of data can be used for Machine Learning purposes as well.



Graph 2 - Effort savings of synthetic data generation using AI and ML vs Manual TDM

## 8 Conclusion

All said and done, even with advanced AI and ML, these approaches have their own positives and negatives. For a successful TDM implementation it is key that the TDM architect, application teams and other key stakeholders agree on a solution that best fits the needs of the enterprise but keep an open eye for the fast evolving TDM solutions in the AI space.

## References

### Tools:

1. <https://app.grammarly.com> was used to review grammatical correctness in some of the sections.
2. Google slides and docs were used to prepare the paper content.

### Web Sites:

1. Praveen Bagare and Ruslan Desyatnikov. 2018. "Test Data Management in the Software Testing Life Cycle". <https://www.infosys.com/it-services/validation-solutions/white-papers/Documents/test-data-management-software.pdf> (accessed Jun 01, 2024)
2. Christopher Manning. 2020. "Artificial Intelligence Definitions". <https://hai.stanford.edu/sites/default/files/2020-09/AI-Definitions-HAI.pdf> (accessed Jun 24, 2024)
3. "Health Insurance Portability and Accountability Act (HIPAA)". <https://www.hhs.gov/programs/hipaa/index.html> (accessed Jun 08, 2024)
4. "General Data Protection Regulation (GDPR)" <https://gdpr.eu/what-is-gdpr/> (accessed Jun 08, 2024)

# Product Owner in Test

**Srilu Balla (Sridevi Sabarinath Balla)**

<http://www.linkedin.com/in/SriluBalla>

## Abstract

In the ever-evolving landscape of software development, traditional Quality Assurance (QA) practices have become increasingly insufficient, often relegated to narrow scopes within sprint-specific tickets. This constrained approach hampers comprehensive product evaluation and undermines the potential for holistic quality enhancement. Moreover, prevailing methodologies like Agile fail to adequately integrate the QA role, leading to misconceptions of QA professions such as:

1. **Agile Challenges:** Agile methodologies do not recognize the QA role in their frameworks.
2. **Limited Scope:** QA teams are frequently restricted to testing tasks defined in tickets, which do not encompass the entire product or its integration with other systems.
3. **Lack of Empowerment:** Organizations ignore the recommendations made by QA professionals.
4. **Scapegoat:** QA is typically only remembered when something goes wrong.
5. **Bug Finders:** QA is perceived merely as 'bug finders'; some QA teams function as such too.
6. **Final Step:** QA is considered the 'final step in the process'. Not understanding their needs.
7. **Automation Overreliance:** Automating is perceived to be superior to a human testing a product.
8. **Process Confusion:** There is often confusion between testing and acceptance criteria checking
9. **Lack of Standardization:** Bugs and test cases created by one person are hard for others to understand and utilize.

This paper introduces the role of the "Product Owner in Test" (POT), a transformative position designed to revitalize and expand the scope of quality practices in software development. By redefining titles—such as reimagining QA as "Quality Assistant"—and introducing new responsibilities and frameworks, the POT role seeks to address longstanding challenges. Key proposals include:

- **Terminology Evolution:** Reframing "Test Cases" as "Check Cases" to distinguish between exploratory testing and pass/fail validation, and viewing "Bugs" as opportunities for improvement rather than failures.
- **Expanded Roles:** Establishing positions like Product Owners in Test (POTs), Quality Assistants, and Software Developers in Test (SDETs) to ensure a comprehensive approach to product quality.
- **Broadened Testing Scope:** Advocating for an evaluation of the entire product ecosystem, beyond the confines of individual tickets.
- **Standardization and Certification:** Introducing certifications such as Certified Product Owner in Test (CPOT), Scaled CPOT (SCPOT), Certified Issue Reporting Professional (CIRP), and Certified Check-Case Design Professional (CCDP), accompanied by a robust code of conduct.

## Biography

*Sridevi "Srilu" Balla has over 20 years of experience as a Quality Assistant, working in various roles including Quality Professional, Agile Coach, and Product Manager. She developed the Interface Agnostic Automation Framework, which she presented at the PNSQC in 2011 and 2012. Additionally, she presented on the test case life cycle in 2015. Affectionately known as "the American in the Saree," Srilu combines her love for America with her passion for traditional sarees.*

# 1 Introduction

Quality Assurance (QA) has long been an essential part of the software development process. However, much like processes such as Scrum and Agile, QA practices are directly derived from manufacturing methodologies.

In manufacturing, the production process begins after identifying specific needs and requirements. Following research and development, prototypes are tested by beta customers. Once the product enters the manufacturing phase, the Quality Assurance (QA) team steps in with a set of requirements to evaluate the product based on a documented checklist.

In manufacturing, the primary focus is on the product. An entire role, Quality Control, was created specifically to check the product that has been produced, even if the formula has not changed in years. The expectations in manufacturing are very clear and well-defined.

Software development, on the other hand, is unique with every feature, element, integration, and schema. Unlike manufacturing, where products are mass-produced in batches with proven requirements, software is developed one time only, often without extensive R&D or prototyping. Consequently, traditional manufacturing processes have been inadequate for software development for decades.

# 2 New Role and Vocabulary

Repeating the manufacturing life cycle steps in the Software Development Life Cycle (SDLC) will not lead to improved outcomes. To address the gaps and shortcomings in applying manufacturing processes to software development, we should have a unique process to better suit the software development context.

## 2.1 Product Owner in Test

Similar to a product owner who is empowered to add value and make independent decisions, a **Product Owner in Test** (POT) will learn, explain, and reflect on the product. They will own the product's quality and integrity. Teams must collaborate with the POT to include, enhance, or remove any features at any stage of the product life cycle. (Qualifications, Responsibilities listed ahead). One POT per Product. POT works independent of an engineering team.

### 2.1.1 Quality Assistant

Quality Assurance (QA) is often misconstrued as a guarantee of perfection. The role may be accurately described as Quality Assistance. As a **Quality Assistant (QA)**, our focus is on bettering the product rather than guaranteeing specific outcomes. QA works within an engineering team. QAs are skilled professionals who excel in writing standardized test cases and logging detailed bug reports.

While the POT focuses on the broader, strategic aspects of product quality, the Quality Assistants will continue to be invaluable in handling the day-to-day execution of test cases and maintaining the operational aspects of quality assurance. This redefined role allows for a more efficient division of responsibilities, ensuring that both strategic oversight and meticulous execution are covered within the development process.

### 2.1.2 Scaled Product Owner in Test

The introduction of a Scaled Product Owner in Test (Scaled POT) becomes essential as organizations adopt frameworks like SAFe (Scaled Agile Framework) and Scaled Scrum Teams. In this role, the Scaled POT will coordinate with various Scrum teams to ensure that quality assurance practices are aligned across the board. They will be responsible for integrating testing strategies into the broader agile framework,

ensuring that all teams adhere to standardized processes while still addressing the unique needs of each project. This includes overseeing cross-team dependencies, facilitating communication between teams, and ensuring that all products meet the high standards required for successful integration and release.

## 2.2 Testing

Testing is about understanding HOW something works. The only true form of 'testing' is exploratory testing, which is performed by humans, with or without the aid of tools. Unlike checking, testing can extend beyond the requirement scope. It uncovers dependencies, issues, inconsistencies in features, logic, UI/UX, ADA functionality, vulnerabilities, and much more.

### 2.2.1 Checking

Any process focused on PASS/FAIL criteria is checking, not testing. Unit Testing, Integration Testing, System Testing, Acceptance Testing, Performance Testing, Security Testing, Usability Testing, Compatibility Testing, Regression Testing, Smoke Testing, Alpha Testing, and Beta Testing are primarily used to check the known state of a system.

### 2.2.2 Check Cases

Testing reveals how a feature works. Once this understanding is achieved, reliable and repeatable steps can be established. These steps will be the **Check Cases**, which will help verify the application's state.

### 2.2.3 Automated Checking

Automation is a tool that executes tasks as instructed by a human. It cannot perform testing on its own; it can only check for the specific state of the system the code was written for. Therefore, it should be referred to as Automated Checking.

## 3 The Role and Responsibilities of Product Owner in Test

The Product Owner in Test (POT) is a crucial role within an organization, focusing solely on the product's quality and functionality. Unlike traditional QA roles embedded within scrum teams, the POT operates independently, ensuring that the product meets the highest standards before release.

### 3.1 Organizational Level Responsibilities

The Product Owner in Test (POT) plays a critical role in advocating for and implementing high-quality standards across the organization. By focusing on quality at the top level, the POT helps reduce issues throughout the process and products, leading to optimized time management and resource utilization.

#### 3.1.1 Production Data Sanctity

- **Customer Data Protection:** Ensure that customer-sensitive data is never used in testing unless explicitly requested under special circumstances.
- **Data Scrubbing:** All customer and employee data must be thoroughly scrubbed if it is to be used in lower environments to maintain data privacy and security.

### 3.1.2 Standardizing Common Features

- **Feature Uniformity:** Standardize common functionalities such as email formats, password criteria, registration forms, and authentication processes across all applications.
- **Efficiency in Decision-Making:** By standardizing these features, the POT reduces the decision-making burden on teams, leading to fewer issues and decreased testing effort.

### 3.1.3 Standardizing Lower Environments

- **Clear Environment Designations:** Establish clear designations for environments, including Production, Staging/Demo, QA/Test, and DEV/Sandbox.
- **Organizational Uniformity:** Ensure that the usage and naming of these environments are consistent across the organization, allowing for easy adaptation by any team member

### 3.1.4 Standardizing Releases

- **Release Timelines:** Set up clear timelines for major releases and hotfixes to ensure consistency and predictability.
- **Naming Conventions:** Bring uniformity to naming conventions for fix versions, labels, and tagging processes for release tickets across all products.
- **Code Freeze Protocols:** Define code freeze timelines for all types of releases, ensuring that these practices are consistent across the organization.

### 3.1.5 Standardizing MVP 1 Requirements

- **Foundational Elements Setup:** Ensure that critical foundational elements, such as database setup, API setup, environment setup, and release processes, are established early in the development process.
- **Preventing Rushed Implementations:** Enforce these standards to avoid rushed or incomplete setups that can lead to complications later in the product lifecycle..

## 3.2 Product Level Responsibilities

The role of Quality Assurance (QA) has often been misunderstood and undervalued, with QA teams frequently being seen as mere "bug catchers" who are only brought into the process when something goes wrong. This approach limits the impact QA can have on delivering a high-quality product. To maximize the benefits of the testing process, organizations need to provide the right tools, resources, and responsibilities to the Product Owner in Test (POT). The following recommendations are designed to empower the POT to succeed in this role:

### 3.2.1 Testing the Product

The **Product Owner in Test (POT)** is responsible for overseeing a product in its entirety, ensuring that all aspects of its quality, functionality, and performance are thoroughly evaluated and optimized. This singular focus allows the POT to make informed, strategic decisions based on a deep understanding of the product's complexities and nuances. Key responsibilities include:

- **Comprehensive Testing:** Most of the POT's time should be dedicated to conducting thorough testing across all aspects of the product.
- **Product Overview:** The Product Owner should provide the POT with a thorough understanding of the product, going beyond just assigning tickets for testing.

- **Technical Walkthroughs:** Development leads, architects, and database administrators should provide the POT with detailed walkthroughs of unit tests, integration tests, component tests, and data rules.
- **Access to Environments:** The POT should have access to the product, database, API, and provisioned users or accounts in lower environments, or the knowledge to start testing independently.
- **Strategic Testing Initiation:** Testing should begin only after a significant portion of the product is developed.
- **Holistic Testing Approach:** Testing should be conducted with and independently of the requirements, ensuring a comprehensive understanding of the product.
- **Consistent Testing:** Testing should be a continuous, exploratory process rather than just checking bits and pieces from time to time.

### 3.2.2 Documenting the Product

- **Creating Documentation:** The POT should create detailed check cases that include data, API, and integration steps along with UI steps, document any issues that need resolution, and maintain wikis to help team members understand the product.
- **Clear Test Cases:** Test cases should be written in such a way that anyone who can speak English can execute them and understand the results. The steps should also help identify the root cause of issues.

### 3.2.3 Demonstrating the Product

- **Product Demo:** The POT should conduct a product demo with the engineering and product teams, with or without stakeholders, to showcase the product's features and functionality.
- **Execution of Test Cases:** During the demo, the POT should execute every test case to demonstrate the product's features inside and out.
- **Explaining Data Handling:** The POT should explain how data is stored, processed, and reused within the application.
- **Assessing Product Readiness:** This demo will help determine the product's readiness and overall health.

### 3.2.4 Recommendations and Delegating Tasks

- **Task Delegation:** For large applications or projects, the POT may delegate tasks to Quality Assistants but should always oversee the overall quality of the entire product.
- **Recommendations for Testing:** The POT may recommend additional testing types such as performance, security, or functional automation, and allow the team to make the decision.
- **Enhancing Testing Automation:** The POT should recommend specific unit, integration, component, and functional tests to be added to increase automation, thereby reducing the need for redundant manual checks.

## 3.3 Tools and Resources for POT Success

To ensure the **Product Owner in Test (POT)** is set up for success, the organization should provide the following essential tools and resources:

### 3.3.1 Access to the Product and Database

- **Full Product Access:** Ensure the POT has full access to the product in a clean, dedicated environment for thorough testing and quality assurance.

- **Database Access and Training:** Provide access to a functional database, along with the necessary training to allow the POT to create and manage data independently.
- **Codebase Access:** Grant direct access to the codebase, enabling the POT to gain a deep understanding of the product's inner workings and to verify aspects of the product as needed.

### 3.3.2 Comprehensive Information

- **List of Services and Applications:** Provide a detailed list of all services and applications involved in the product, giving the POT a complete picture of the product's ecosystem.
- **Provisioned Users or Accounts:** Ensure the POT has access to provisioned users or accounts specifically for testing purposes, enabling realistic and thorough testing scenarios.

### 3.3.3 Product Overview

**Product Walkthrough:** A comprehensive walkthrough of the product by the Product Owner or Project Manager is crucial. This should cover all features and functionalities, giving the POT a holistic understanding of the product.

## 3.4 Testing results from POT

### 3.4.1 Test Data Set-up

- **Standardized Test Data:** Establish standardized test data that is accessible and usable by the entire team across all environments, including development, QA, staging, and production.
- **Comprehensive Wikis:** In cases where test data cannot be pre-created, develop detailed wikis with step-by-step instructions for team members to generate their own test data.
- **Consistent Naming Convention:** Implement and enforce a uniform naming convention for all test data to ensure clarity and uniformity across all testing activities.

### 3.4.2 Check Cases

- **Designing Check Cases:** After thoroughly testing and understanding the product, the POT designs and documents product-based check cases.
- **Linking to Resources:** Check cases will include links to wikis where relevant data can be accessed or created.
- **Clear Execution Guidelines:** These check cases should be so well-defined that anyone reading them can easily execute them without needing additional guidance, ensuring consistency and accuracy in testing across the team.

### 3.4.3 Issues / Bugs

- **Nature of Bugs:** Bugs are an inevitable byproduct of testing. Some bugs require immediate attention, while others can be addressed later or may not need to be fixed at all.
- **Clear Issue Logging:** Issues should be logged clearly with detailed steps, properly tagged, and prioritized with input from the team.
- **Opportunities for Improvement:** Issues should be seen as opportunities for improvement, not as indicators of the application's failure. This mindset promotes continuous improvement and helps maintain a positive approach to quality assurance.

## 3.5 Product Release Responsibilities

Once testing is completed and check cases are written down and features have been signed off on, products may be promoted to environments – Stage, PROD. The POT will assist the team in a Go/No-go decision. POT will seldom make that decision.

### 3.5.1 Release Preparation

- **Advance Notification:** The POT ensures that all stakeholders are notified in advance of the release dates and the features targeted for release.
- **Comprehensive Product Check:** Before the release, the POT conducts a final check to ensure all aspects of the product are covered, verifying that the product is ready for deployment.
- **Documentation and Demonstration:** The POT documents any findings from the final checks and demonstrates the product to the team to assist in making a Go/No-Go decision.

### 3.5.2 Production Verification

- **Assisting in Verification:** The POT assists in verifying the product in the production environment to ensure a smooth release process.
- **Data Integrity:** It is crucial that no customer production data is used during testing without explicit consent, and the POT ensures compliance with this standard.

## 4 Qualifications and Skills Required

To successfully lead a product through to production requires a refined skill set. Here are some key competencies that are recommended for this role.

### 4.1 Five years of Experience

- **Minimum Experience:** A minimum of 5 years of experience in a Quality Assurance or Developer role.
- **Software Development Expertise:** Deep understanding of software development processes and methodologies.
- **Beneficial Experience:** Experience as a domain expert or product owner is beneficial, but not essential for this role.

### 4.2 Understanding Software

- **Software Architecture:** The POT must possess a comprehensive understanding of software architecture, design principles, and development practices.
- **Component interaction:** This includes in-depth knowledge of how various components of the software interact and function together.
- **Impact Awareness:** Additionally, they should have a clear understanding of the impact of code changes on the overall system, with a strong emphasis on the significance of data flow throughout the application.

### 4.3 Demonstrating Expertise

The POT should be capable of taking full ownership of the product and effectively demonstrating its functionality through various means such as demos, detailed check cases, documented issues, and comprehensive wikis.

#### 4.3.1 Product Demo - How it works

- **Execution and Demonstration:** The POT should be able to demo the product by executing it in front of the team as needed.
- **Comprehensive Presentation:** This includes demonstrating test cases, describing issues, presenting stage and production releases, and showcasing the product to stakeholders.
- **Stakeholder Understanding:** This hands-on approach ensures that everyone involved has a clear understanding of the product's current state and its readiness for release.

#### 4.3.2 Check Cases

- **Designing Check Cases:** The POT must excel at designing product-based check cases, created after thoroughly testing and understanding the product.
- **Clarity and Execution:** These check cases should be well-defined, ensuring that anyone reading them can easily execute them without additional guidance, thereby maintaining consistency and accuracy in testing across the team.

#### 4.3.3 Data Setup

- **Data Proficiency:** The POT must be proficient in setting up data for both current and future checking needs, ensuring all necessary scenarios are covered.
- **Comprehensive Wikis:** Additionally, they should maintain detailed wikis that guide the team on effectively using and creating data, providing a valuable resource for consistent and accurate testing practices across the organization.

### 4.4 Programming Knowledge

- **Understanding Developers' Work:** By having a basic grasp of programming, the POT can better understand the challenges and approaches developers take.
- **Informed Testing Decisions:** A POT with programming knowledge can make meaningfully and effectively understand the application and its code.
- **Debugging Skills:** While the POT is not expected to be a full-fledged developer, having basic debugging skills is important. It enables the POT to collaborate more effectively with developers by offering insights into potential problems and assisting in resolving issues more efficiently.

## 5 Product Ownership in Test Across Multiple Products

A POT is focused on a single product, the role can be expanded to manage multiple products, especially in organizations that have a suite of related products or where the products are not highly complex.

### 5.1 Importance of Dedicated POT for Each Product

- **Clear Ownership:** Each product should have a dedicated POT to maintain clear ownership and responsibility. This ensures that there is no ambiguity in decision-making and that the product's quality assurance is guided by a single vision.
- **Avoiding Conflicts:** When multiple POTs are responsible for the same product, it can lead to conflicting decisions, inconsistent testing strategies, and potential delays. A single POT ensures that all decisions are aligned with the product's goals and quality standards.
- **Streamlined Communication:** A single POT per product simplifies communication channels with the development team, stakeholders, and other involved parties, leading to more efficient collaboration and quicker resolution of issues.

## 5.2 Challenges of Managing Multiple Products

- **Time Management:** A POT managing multiple products must be highly effective in time management to ensure that each product receives attention and that deadlines are met.
- **Balancing Priorities:** The POT must constantly balance the priorities of different products, which may have varying levels of complexity, risk, and importance to the organization.
- **Maintaining Quality:** The POT must ensure that the quality of each product does not suffer due to the divided focus. This requires a strategic approach to testing, automation, and continuous monitoring.

## 6 Not Responsibilities of a POT

Last but not least, it is important to delineate the boundaries of this role to avoid misunderstandings about what the POT is and is not responsible for. Certain aspects of the development process remain the collective responsibility of the entire team.

### 6.1 Not a Bug Finder

While the POT will inevitably identify issues during the testing process, it is crucial to understand that finding bugs is not the primary responsibility of this role. Bugs are a natural byproduct of the testing process. Their discovery is a shared responsibility across the entire team, including developers, testers, and other stakeholders. The POT's focus is on understanding and improving the product as a whole, not just on the identification of individual defects.

### 6.2 Not Responsible for Validation of Acceptance Criteria

Validation of acceptance criteria is a critical part of the Scrum process, but it is not the responsibility of the POT. This task falls squarely within the domain of the Scrum team, which includes developers, testers, and the Product Owner. The POT's role is more strategic, focusing on the broader picture of product quality rather than on the specifics of individual user stories or features.

### 6.3 Not responsible for Overall Quality of the Product

Ensuring the overall quality of the product is a team effort and cannot be the sole responsibility of the POT. Quality is a collective goal that requires input and collaboration from all team members, including developers, designers, testers, and the Product Owner. The POT's role is to provide insights and recommendations based on their testing, but the implementation of these recommendations is a shared responsibility.

### 6.4 Not Decision Maker of the Product

Decision-making regarding the product's direction, features, and priorities is the responsibility of the Product Owner and the Scrum team. The POT's role is advisory, providing feedback based on their testing experience and insights into the product's performance. However, the final decisions about what features to develop, modify, or remove rest with the Product Owner and the team.

### 6.5 Not Gatekeeping Releases

The POT is not a gatekeeper who has the authority to block or approve releases. Instead, their role is to provide valuable feedback and insights to help the team make informed go/no-go decisions. The POT may highlight potential risks or issues, but the decision to proceed with a release should be made collectively by the Product Owner, Scrum team, and other relevant stakeholders.

## 6.6 Not a Domain Expert

The POT does not need to be a domain expert; their primary focus is on software and how it functions with the given requirements, rather than deep subject matter expertise in the product's specific domain.

## 6.7 Not a QA manager

When the POT delegates tasks such as performance testing, security testing, and regression testing, it's important to understand that this does not equate to a QA managerial role. The POT's focus remains on ensuring the product's quality and integrity, and delegation is simply a means to efficiently allocate specialized tasks to those best equipped to handle them.

# 7 Conclusion

In today's complex and rapidly evolving software development landscape, the traditional roles and practices of Quality Assurance (QA) are no longer sufficient to meet the demands of ensuring product quality. The introduction of the Product Owner in Test (POT) role represents a significant shift towards a more integrated, holistic approach to product quality. By focusing on the overall functionality, integrity, and quality of the product, the POT ensures that testing goes beyond simply validating requirements to providing a deeper understanding of how the product truly works.

Throughout this paper, we explored the limitations of conventional QA practices and the need for new roles, such as the POT and Quality Assistant, to better align with the unique challenges of software development. We discussed how the POT's responsibilities, including comprehensive testing, documentation, product demonstration, and collaboration across teams, position them as a crucial figure in the software development process. Moreover, the introduction of the Scaled Product Owner in Test (Scaled POT) role addresses the complexities of larger organizations adopting frameworks like SAFe, ensuring consistent quality standards across multiple teams and projects.

The proposed changes in vocabulary, such as distinguishing between testing and checking, further reinforce the need for a paradigm shift in how we approach software quality. By redefining the responsibilities and expectations of QA roles, and embracing the POT's strategic focus, organizations can better navigate the intricacies of modern software development. This approach not only leads to the delivery of higher-quality, more reliable products but also fosters a culture of continuous improvement, collaboration, and excellence in the development process.

As we move forward, the **Product Owner in Test (POT)** will play a pivotal role in shaping the future of quality assurance in software development, ensuring that products are not only functional but also meet the highest standards of security, performance, and user experience.



# Achieving Software Quality Excellence Through Software-Hardware Co-Design and Co-Verification

**Monica Bao, Sophia Lee, Havish Sripada, Joanna Mei**  
**[revampedrobotics@gmail.com](mailto:revampedrobotics@gmail.com)**

## Abstract

The future of technological advancement and complexity increasingly emphasizes the importance of software-hardware co-design and co-verification, a process currently practiced in the field of embedded systems, especially robotics. This paper details the experiences of a FIRST (For Inspiration and Recognition of Science and Technology) Tech Challenge (FTC) robotics team, RevAmped Robotics, in implementing various software-hardware features to enhance a robot's user-controlled and autonomous performance.

This paper presents software-hardware co-design and co-verification methodologies to improve robot performance, usability, and optimization. The team strategically incorporated a diverse array of independently operating sensors to automate the robot and enhance its functional capability in both the user-controlled and autonomous periods. The sensors, built into different robot components, are processed with parallel executing software threads, each of which transmits the information to a different color on an LED panel. This allows the robot to communicate its state to the user in real time, allowing for simple testing of robot features and simplifying user-based decision-making. The team also successfully synthesized complex, self-adaptive localization techniques and path-generating algorithms such as Pedro Pathing to optimize the robot's automated performance. In addition, using an automated tester program allowed the team to quickly identify, isolate, and correct any sensor, motor, or software errors the robot encountered. The team improved the software quality of the robot by accurately controlling its components, communicating its state to the user in real time, optimizing the automated movements, and having an automated testing system. This is a significant step forward in demonstrating the importance of software-hardware co-design and co-verification in improving the software quality of any embedded system.

## Biography

*The authors are high school students from the FIRST Tech Challenge team RevAmped Robotics based in Portland, Oregon. The team has been competing for over seven years at the local and international level and represented Oregon in the 2024 World Championships. The team also aims to learn from industrial professionals to improve their engineering skills.*

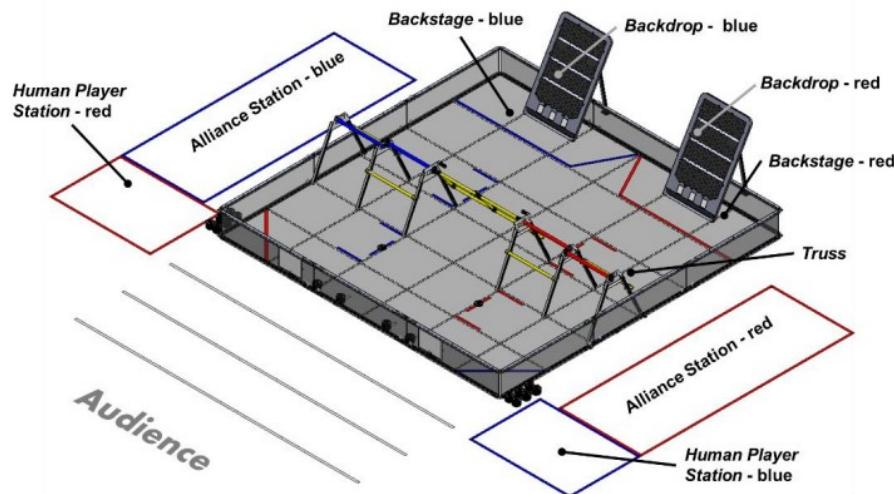
# 1. Introduction

It has long been recognized that experiential and hands-on education provides superior motivation for learning new material by providing real-world meaning to otherwise abstract knowledge. Robotics is an effective tool for hands-on learning, not only of robotics itself but of general topics in Science, Technology, Engineering, and Mathematics (STEM). Learning with robotics gives students an opportunity to engage with real-life problems that require STEM knowledge. FIRST is among the broad spectrum of avenues for pursuing robotics at the pre-university level to promote STEM worldwide. FIRST stands for "For Inspiration and Recognition of Science and Technology" and is an international youth organization focused on developing ways to inspire students in engineering and technology fields [1].

The FIRST Tech Challenge (FTC) is designed for students in grades 7–12 working in teams to compete with each other on a playing field. Teams are responsible for designing, building, and programming their robots to compete in an alliance format against other teams. The robot kit is programmed using Java. Teams, including coaches, mentors, and volunteers, are required to develop strategies and build robots based on sound engineering principles. The ultimate goal of FTC is to reach more young people with an accessible opportunity to discover the excitement and rewards of STEM.

## 1.1 Background for FIRST Tech Challenge Robot Programming

The average FTC season is 9 months long, with a maximum of 15 team members. The FTC competition field is 12' x 12'. Each match is played with four randomly selected teams, two per alliance. Four 18" x 18" robots must be able to navigate around each other without breaking when hit by another robot.



*Figure 1. FTC field setup for the 2023-2024 FTC season [2]*

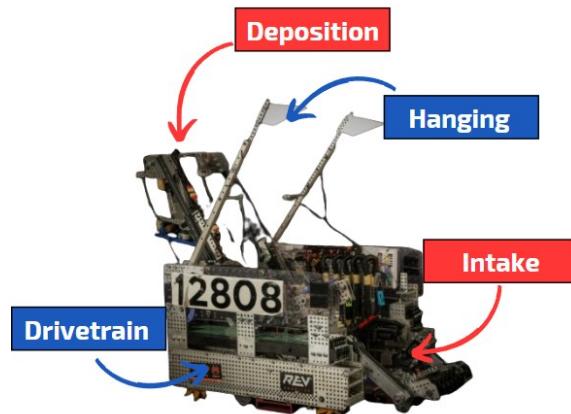
Figure 1 shows the field setup for the 2023-24 FTC season. The FTC robot game comprises two phases: (1) the autonomous phase and (2) the driver-controlled teleoperation phase. The robot is preloaded with a yellow and purple tile called a "pixel," and a team-created "Team Prop" is placed on the field. During the Autonomous period, the robot is expected to place the purple pixel on the randomized location of the Team Element and the yellow pixel at a randomized location on the backdrop. During the driver-controlled period, the goal is to collect as many pixels as possible to deposit on the backdrop. Additional points are awarded when patterns of tiles are placed on the backdrop. During the last 30 seconds of the game, the

robot is able to gain additional points by launching a paper airplane called a “drone” and fully hanging on the frame of the field.

## 1.2 Contributions

The FTC robot implements many advanced industrial technologies on a smaller scale, making it an excellent case study for testing many different technologies. Precise control of the robot and knowledge of its state at all times is crucial for fulfilling the many requirements of an FTC match. Robotic software must sustain accurate hardware control while presenting an intuitive interface for the user and ensuring high system performance. In addition, rapid robot improvements require an efficient and robust testing process.

The team achieved this by implementing many technologies into our robot through an LED panel display, as seen in Figure 2. The robot is acutely aware of where it is on the field, using ultrasonic distance sensors, AprilTags [10], and odometry dead wheels. Finally, advanced trajectory generations are used to improve the reliability of the robot’s automated path-following. Automated testing of the robot’s components is needed for rapid iterations and maintenance during competitions.



*Figure 2. RevAmped Robotics’ robot for the 2023-2024 CENTERSTAGE FTC Season*

## 1.3 Software-Hardware Co-Design and Co-Verification in FTC robots

Software-hardware co-design [5] is the concept where both the software and hardware of an embedded system are designed simultaneously to maximize the system’s capabilities. For embedded systems, trade-offs need to be made between power consumption, materials, time, complexity, and capability. Hence, designing a full system and separating the system into hardware and software components becomes essential. The same is true for all robotics.

Software-hardware co-design and co-verification [6] are pivotal in robotics due to the integration of mechanical, electronic, and computational components. Co-design involves the simultaneous development of hardware and software, ensuring that both components are optimized to work seamlessly together. This process begins with defining the robot’s requirements and partitioning tasks between hardware (such as motor control and signal processing) and software (such as decision-making algorithms and path planning). After implementing both hardware and software components, co-verification techniques are employed to continue iterating upon the robot.

Co-verification ensures that the integrated hardware and software function correctly, employing comprehensive test benches, integrated testing environments, and advanced debugging tools. These

methodologies' benefits include optimized performance, reduced development time, cost efficiency, and enhanced reliability and quality of robotic systems. By addressing these challenges, hardware-software co-design and co-verification enable the development of high-performance, adaptable, and reliable robotic systems capable of performing complex tasks in diverse environments.

This paper presents case studies and techniques that the team employed to follow software-hardware co-design and co-verification principles in order to build a high-functioning robot. An automated tester code quickly verifies the validity of our robotic prototypes while the team implements localization and path generation systems to increase the autonomous capabilities of our robot. Combined with sensors to monitor the state of the robot constantly, the utilized principles of software-hardware co-design and co-verification iterate upon the robot over the span of the FTC season. One strategy the team uses to employ software-hardware co-design and co-verification is our design cycle, composed of brainstorming, prototyping, fabricating, and testing. Details about the cycle are shown in Figure 3.

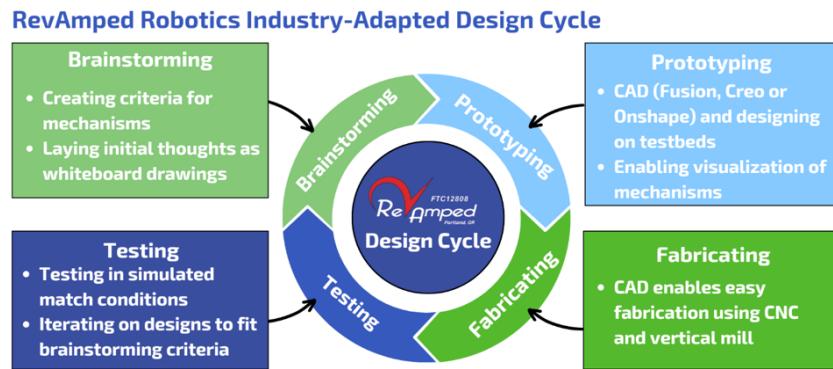


Figure 3. System Design Cycle

### 1.3.1 Deposition Example

One example of using software-hardware co-design and co-verification is the design of the deposition of the robot, where its finalized design was produced through two iterations of the RevAmped design cycle.

Starting with brainstorming, the team created a whiteboard drawing of their initial ideas, where they concluded that a drop-box style deposition would have the potential to be explored. Computer-aided design (CAD) plays a vital role in prototyping by allowing us to create models and simulations of each part without wasting physical materials. Each deposition design is entirely made in CAD and tested to work with other parts. To enhance our initial latch, the team wanted to prevent both pixels from depositing out at once. To do this, a stub was added to our new latch design, giving us more control while creating mosaics – a specific pattern of pixels that score additional points when created. The team decided to program different angles that the latch could be at to give us the flexibility to drop pixels onto the backdrop one at a time or all at once.

The team realized an advanced design with a more extensive range of movement was necessary. Once again, using CAD, the team designed the new parts for the deposition into two rotatable claws. After multiple iterations, we created a claw mechanism that gave us a smooth intake to deposition transfer. With 180 degrees of motion, the claw significantly increased our precision and speed when creating mosaics. This design had a total of three servos, and the use of programming software designed to synchronize the three servos allowed for a much more efficient deposition system. Figures 4 and 5 illustrate the changes in deposition system.

<b>Initial Deposition</b>	
<b>Side Dropping Box</b> Simple box mechanism, with two doors and pixels dropping side to side.	
<b>Final Front Drop Box</b> Dropped pixels from the front using one servo, and became more compact.	
<b>Initial Latch</b> Consisted of a bar and cover to block pixels from falling.	
<b>Final Latch</b> Enabled the option to drop one or two pixels at once.	
<b>Software Iteration</b>	
The first deposition system has one servo controlling one latch. By raising the servo incrementally, we investigated two pre-set servo positions. The first servo position would drop the pixel closest to the intake opening. The further servo position would drop the pixel further from the intake opening.	
	

Figure 4. Initial Deposition

<b>Revamped Deposition</b>	
<b>Claw Iteration 1</b> Consisted of two small claws connected by a gear that grabbed the pixel from the inside and outside. (Required too much precision to grab pixels)	
<b>Mounting Iteration 1</b> Consisted of two servos connected to two claws, allowing us to grab two pixels. (Too flimsy and weak)	
<b>Current Claw</b> Consisted of rubbery noodles allowing grip. Thicker claw arm with 2 stationary points of contact, requiring less precision.	
<b>Current Mount</b> Uses one servo to give 180 degrees of rotation, aiding in mosaic creation, with 2 claws gripping accurately into the pixel	
<b>Software Iteration</b>	
Our deposition system went from using one servo to using three servos. Two servos are used to grab and drop the pixels, while the third servo is able to tilt the axis of deposition. By coding three different preset axis that the deposition can be in, we are able to deposit pixels for any arrangement that the backdrop is in. We also coded our software to be able to drop either one or both pixels on to the backdrop.	
	

Figure 5. Revamped Deposition

## 2. Automated Testing

Automated tester code is highly beneficial for embedded systems due to its ability to provide consistency and reliability. Automated tests ensure that the robot behaves and functions correctly under a variety of circumstances to test the system's abilities. These tests require less labor than manually testing every component of an embedded system, allowing for rapid identification of problems and quick development iterations.

Automated tests are also crucial for regression testing, ensuring that new changes do not negatively impact existing functionalities. This is particularly important in complex systems where parts need to be replaced and recalibrated, as the automated tester code can catch if the new part doesn't integrate well into the system. Furthermore, automated tests serve as documentation for the expected behavior of the embedded system's software, providing a transparent and traceable record of what has been tested and the outcomes, which aids in debugging and future development.

### 2.1 Automated Testing in FTC

When playing multiple games, quick robot maintenance is critical to making sure our robot is able to operate at its full potential. By having an automated tester code, our team is not only able to ensure our robot is functioning without issue, but our team can also quickly detect the core reason for any

malfuction the robot is receiving. Our tester code is able to test each motor, sensor, and servo on the robot individually for functionality, allowing the team to identify if the problem faced is a hardware or software error. If the malfunction is a software error, the team will be able to identify the specific component that needs to be fixed quickly.

We are able to use our tester code for multiple functions at once. For example, two types of sensors on the robot are tested at the same time. Ultrasonic (distance) sensors and limit switches are both tested at once, which is something that cannot be done by manually testing sensors one at a time.

Sensor testing is also automated. For example, limit switches behind horizontal and vertical slides ensure that the slides don't over-retract and burn the motors powering the slides during the match. The program automatically tests the limit switches (force sensors) by extending and retracting the linear slides, measuring whether the sensor feels a change in force when the slides are away from the sensor as opposed to weighing down on it. Other sensors, such as voltage sensors and ultrasonic (distance) sensors, are also tested by the program controlling servos and motors and checking if the sensors respond correctly to this change.

*Table 1. Tester Code Functions*

Motors	Servos	Sensors
Drivetrain; Horizontal Slides; Vertical Slides; Intake Motor.	Intake Door; Door; Intake TwoBar; Drone; Droppers; Hang; Deposition TwoBar; Deposition Turner; Deposition Claws.	Ultrasonic Sensors; Voltage Sensors; Encoder Sensors; IR sensor; LED Panel.

This automated tester is a highly versatile tool for testing the robot quickly. Whether it's between matches during a competition, during regular maintenance of the robot, or as a confirmation that everything works as expected after making mechanical changes to the robot, it's able to identify issues and offer servo calibration tools. This eliminates the need to test every individual component of the robot manually and instead allows groups of motors, servos, and sensors to be tested at once.

### 3. Case Studies

#### 3.1 Case Study: Intake Sensors + LED Panel

Through the season, each of the robot's mechanical components will undergo continuous development and refinement through a synergized software-hardware co-design and co-verification agile process. This ensures not only precision but also optimal functionality of its mechanisms. Illustrated through the exemplar of the intake system, the team has strategically incorporated a diverse array of sensors, notably leveraging the capabilities of an IR sensor to its fullest extent. Furthermore, in pursuit of enhanced operability for the robot's drivers, the team has harnessed software tools to streamline controls.

However, at the beginning of the season, our intake contained no sensors. Instead, we relied on a time-based system with pre-programmed wait-time functions to collect two pixels, the maximum amount allowed in a robot, for each cycle. While they temporarily worked, the team realized that the wait-time functions had multiple flaws. A significant issue was how the functions would sometimes cause our robot to leave the pixel stack too early or late. This led to instances where the intake would collect three or even no pixels before moving on to the following autonomous function. Such inconsistencies caused our autonomous cycles to become unreliable and resulted in a decrease in potential points. To help minimize error and maximize efficiency, the team considered using an IR sensor to detect the number of pixels in our intake.

Before incorporating the IR sensor into our robot, the team had to confirm whether or not the IR sensor would ultimately aid the robot's design more than the wait-time functions. To do this, the team utilized SWOT analysis [3] [4] to evaluate which option held the most promise. Following the acronym SWOT, the team wrote down the Strengths, Weaknesses, Opportunities, and Threats of both potential choices in two tables. After comparing the tables, the team concluded that the IR Sensor would lead to a more significant overall benefit and was mounted to the side of our intake for easy pixel detection.

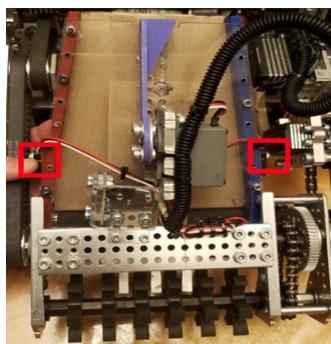
*Table 2. SWOT Analysis for Pre-Programmed Wait-Time Functions*

STRENGTHS	WEAKNESSES
* Allows our intake to collect two pixels * Doesn't take up space in the robot	* Functions aren't always on time
OPPORTUNITIES	THREATS
* No hardware maintenance	* A lot of time spent on fine-tuning wait time * Losing points * Possibility of penalties

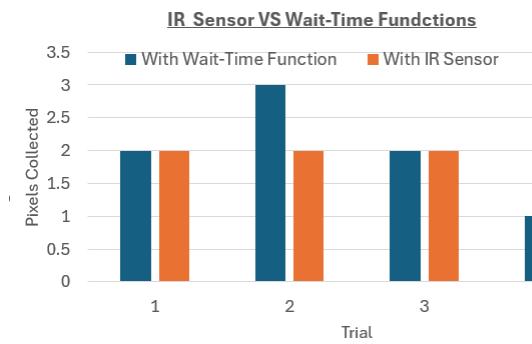
*Table 3. SWOT Analysis for IR Sensor*

STRENGTHS	WEAKNESSES
* Constantly detects the number of pixels in our robot * Very reliable * Simplify the program	* Increased power consumption * Depleted battery power quicker
OPPORTUNITIES	THREATS
* Consistently collects two pixels every cycle * Efficient autonomous cycling * Maximize points	* Not enough time to mount before the next competition

To visually compare the IR sensor with the wait-time functions, the team decided to code and run a few trials of two separate programs for the robot's autonomous phase: one for the wait-time functions and the other for the IR sensor. Figure 7 shows the results for each trial and compares how accurate each option was when intaking pixels for each trial. Looking at the graph, you'll notice that the IR sensor is much more precise and stable; every run collects two pixels, while the wait-time functions fluctuate with each trial. This not only proves that the wait-time functions are inconsistent and unreliable but also proves that the IR sensor is much more consistent.



*Figure 6. IR sensors on the robot's intake*



*Figure 7. Comparison of wait-time functions and IR sensor accuracy*

Additionally, after adding the IR sensor to the robot's design, we implemented an LED panel onto the side of our robot. Using the output from our IR sensor during the autonomous period, our LED panel lights up a specific color based on preprogrammed commands. These colors always indicate the number of pixels in our robot. This made in-game decisions for the robot's drivers more accessible and helped maximize scoring capabilities and efficiency when gathering pixels.

## 3.2 Case Study: Robot Movement

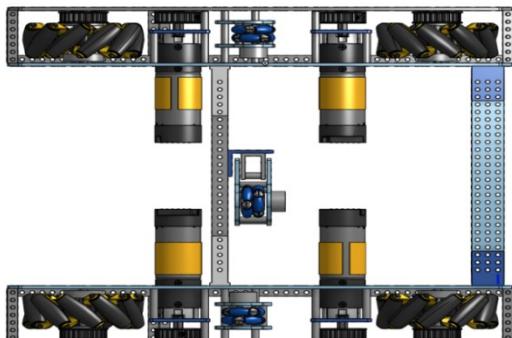
### 3.2.1 Introduction to Path-Following

Path-following has long been an essential field in the application of unmanned vehicles (UVs). The ability of such an autonomous device to determine its precise position and orientation on the coordinate field, referred to as localization, is critical to a robot's precise traversal of a given curve. This self-characterization is thus utilized in a process called trajectory generation, the process the robot uses to determine where it wants to move next. To further maximize the robot's capability, the implementation of various industry-adapted localization methodologies improved the quality of the robot's pursuit of its encoded autonomous trajectories. The exemplar of the First Tech Challenge robot, which must possess the capacity to adapt to changes in its environment, powerfully illustrates the power of software-hardware co-design and co-verification when undertaking complex tasks.

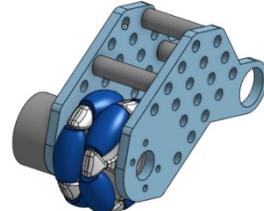
### 3.2.2 Mecanum Drivetrain and Odometry

In order to produce the highest accurate trajectory, robots must include the successful synthetization of both hardware and software methodologies. Flexible hardware mechanisms are critical to the robot's ability to make decisions autonomously and traverse the field in the most optimal manner. This condition is especially true in the First Tech Challenge, where robots must compete in a timed environment to rapidly navigate through their surroundings on the obstacle-dense playing field.

The team employed the mecanum drivetrain [9] to facilitate flexibility in the robot's movement mechanism. The following image reveals the various intricacies behind the mecanum drivetrain. Figure 8 shows the layout of the mecanum motors and wheels in the drivetrain. The four mecanum motors are oriented to apply force to two independent axes, allowing for precise holonomic movement [9], such as strafing.



*Figure 8. Bottom View of the Drivetrain Highlighting Mecanum Wheels*



*Figure 9. Odometry Pod*

One example of software-hardware co-design on the robot is the three-wheeled odometry system. This odometry system utilizes dead wheels, omnidirectional wheels that do not immediately contribute to the robot's motion (as shown in Figure 9). Through these miniature dead wheels, the robot can keep track of how far it has moved. The placement of the odometry pods, which carry the dead wheels, is shown in Figure 8. Given that the competition's autonomous cycling mandated back-and-forth movement throughout the y-axis, the team decided to use two dead wheels on the y-axis and one on the x-axis to track the position most optimally. These axes refer to the field coordinate system, where the origin lies at the bottom left of the field and as the robot travels rightward its x-coordinate increases while its y-coordinate increases as the robot moves forward up the field. The coordinate system is measured in inches and spans from the point (0,0) to (144,144).

Using the odometry system, the team was able to calculate the robot's position in a looped parallel thread. Every time the robot moved, the dead wheels would track how far the robot moved by converting the number of wheel rotations to inches of movement.

### 3.2.3 Localization using Kalman Filter

The main co-verification localization algorithm the team implemented in conjunction with the hardware-based odometry mechanism was the Kalman filter [8], a recursive model-based methodology designed to remove noise from a system. This filter allowed the team to utilize other means of obtaining localization output and then merging those separate readings with the odometry values to create a better estimate of the robot's pose. These separate localization readings came from strategically mounted ultrasonic sensors and webcams and allowed further validation of odometry readings.

The ultrasonic distance sensors enabled measuring the robot's position on the field by measuring the distance from the robot to the field wall using the following two equations: the decision between addition and subtraction is based on which sector of the map the robot is on.

$$x = 72 \pm \max(\sin \theta, \cos \theta) \cdot ((D(S, W) + 0.5 \cdot D(R, S) \cdot D(S, E))$$

$$y = 72 \pm \max(\sin \theta, \cos \theta) \cdot D(S, W) \cdot D(S, E)$$

where  $D(S, W)$  is the distance between the sensor and the wall,  $D(S, E)$  is the distance between the sensor and the encoder on the y-axis, while  $D(R, S)$  is the distance between the center of the robot and the sensor, and  $\theta$  is the robot's current heading.

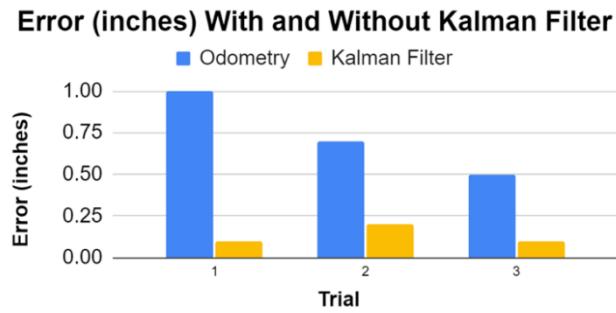
The webcam, mounted at the center of the robot's rear end, was able to provide accurate readings through the use of AprilTags, which are visual images lying above the intaking stacks and on the deposit backdrop for robots to detect. The team calibrated the webcam to utilize these AprilTags to output the robot's distance from the backdrop or stack and then applied the following equations to calculate the robot's pose, the first equation computing the robot's heading relative to the field coordinates and the following two equations computing the robot's distance from the AprilTag in either axis of the field:

$$\begin{aligned} \theta &= 90 \cdot (1 + \operatorname{sgn}(D_x(C, A))) - \arctan(D_y(C, A)/D_x(C, A)) \cdot \operatorname{sgn}(D_x(C, A)) + \\ &\quad \arctan(D_x(C, A)/D_y(C, A)) \cdot \operatorname{sgn}(D_x(C, A)) \\ x &= D_x(C, A) + D(R, C) \cos \theta \cdot \operatorname{sgn}(\theta + \frac{\pi}{2}) \\ y &= D_y(C, A) + D(R, C) \sin \theta \end{aligned}$$

Where  $D_x(C, A)$  and  $D_y(C, A)$  represent the distance in the x and y axes from the camera to the AprilTag and  $D(R, C)$  is the distance from the center of the robot to the camera. The correct equation for the heading is decided based on information from the robot's current orientation in relation to the AprilTag.

It is important to note that the distance sensors were unable to verify the robot's heading, and the algorithm would quickly fall apart if the heading estimate is inaccurate, but the webcam was able to detect the robot's heading by calculating the angle at which the robot viewed the AprilTag. This algorithm allowed the team to improve robot performance within the match further because it gave the robot a solid way to relocalize after collisions with obstacles, which would offset the odometry readings.

The importance of this algorithm can be measured through an experiment involving running the robot back and forth thrice in a straight line parallel to the y-axis. We define the error as the unsigned change of the robot's position in the axis perpendicular to movement after having terminated the motion. Figure 10 displays this error with and without the use of the Kalman Filter. The noticeable difference in errors between the trials highlights the importance of solid localization software-hardware co-verification algorithms.



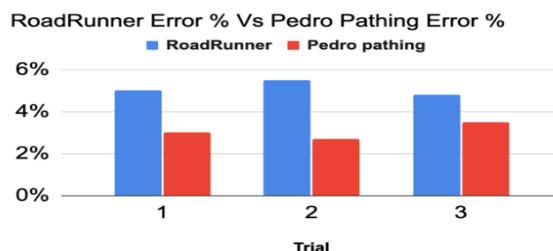
*Figure 10. Comparison of Error With and Without Kalman Filter*

### 3.2.4 Trajectory Generation

Precision and efficiency are crucial to maximizing scoring capabilities during the self-driving, autonomous period. Alongside implementing odometric localization in our robot, various trajectory planning libraries are utilized to ensure the robot achieves near-perfect path-following. To take complete advantage of the specific amount of odometry pods, the co-design capabilities of a new path-following algorithm called Pedro Pathing [7], commonly referred to as Pedro, significantly increased accuracy compared to our previously used system named Roadrunner, a motion profiling library designed for precise control over velocity and acceleration when designing complex autonomous robot trajectories.

Although the odometry pods helped increase accuracy, the critical problem with Roadrunner is that it is a motion profile-based follower. This means that a set of instructions, based on how long to run a certain magnitude of motor power, is calculated for each path before the robot begins following a trajectory. However, Roadrunner uses localization with only the purpose of generating the set of instructions for the robot's motion and, therefore, assumes that the robot is at the correct position at the end of the motion profile, changing its odometry value to match the pre-programmed target value where the trajectory was supposed to terminate. This inadequate application of software-hardware co-design means that Roadrunner is unable to correct itself after the motion profiling sequence, causing discrepancies after collisions or imprecise movement. The robot will continue to be that certain distance off from the target path for the duration of the autonomous period. This season's obstacle-dense playing field meant that substantial error in the robot's position was not sustainable.

To solve the problem of correction after collisions and imprecise movement, the team decided to utilize a more complex path planning library to create their autonomous trajectories. Pedro Pathing is a newly developed open-source system by another FTC team that improves upon the original functions of the widely used First Tech Challenge database named Roadrunner. By dynamically calculating a set of vectors that correct the robot to the closest point on the target curve before proceeding with trajectory-related motion, the Pedro Pathing algorithm allows the robot to detect and correct errors in its position after colliding with an obstacle or following inaccurate motion.



*Figure 11. Comparison of the Percentage of Error for Pedro Pathing and Roadrunner*

Excerpt from PNSQC Proceedings

PNSQC.ORG

Copies may not made or distributed for commercial use

Page 10

To compare the efficiency of both Roadrunner and Pedro Pathing, we developed code that moves the robot in a straight line back and forth for 30 seconds. As seen in Figure 11, multiple trials were run, and Roadrunner had an average end error of 5.1% while Pedro had an end error of 3.1%. We can notice that Pedro has a smaller margin of error and follows a lot closer to the intended path than Roadrunner. This thus proves the importance of software-hardware co-design in improving the accuracy of our robot's autonomous movement.

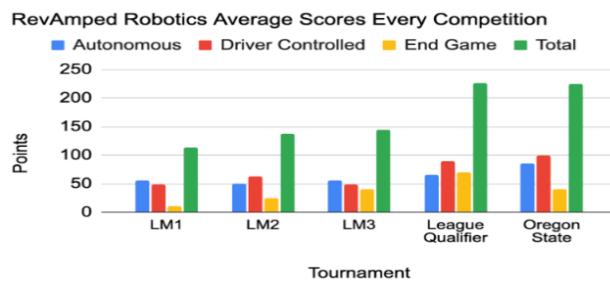
## 4.0 Data Analysis to Drive Future Development

Data analysis on past embedded systems performance, like the analysis done on our robot in Figure 12, helps pinpoint and investigate possible improvements to the embedded system, such as mechanical malfunctions or software bugs. Statistical techniques can then be used to summarize and interpret performance data, often visualized through graphs and charts, making data visualization easier for people. Strategic insights are drawn to identify areas of excellence and needed improvements, informing adjustments in strategy, design, or programming.

This process facilitates a feedback loop where findings are used to iteratively improve the embedded system's design and operational strategy. The benefits of this analysis include data-driven decision-making, a competitive edge, and continuous learning and improvement.

### 4.0.1 Competition Score Analysis in FTC

Over a competition season, many tournaments are played where the team receives feedback on the performance of the robot when competing with and against other robots. By tracking and analyzing competition results, long-term performance trends become evident. By examining performance across several competitions, teams can spot persistent problems or errors that might not be apparent from a single event. This signals to the team what components of the robot work well and possible improvements that could be made. Analyzing competition scores helps teams make data-driven decisions about design modifications, programming adjustments, and strategic changes.



*Figure 12. Average Score Per Competition*

Figure 12 shows the average scores of RevAmped Robotics during every competition. The average scores of the team are broken down into points during the 3 phases of the match, the autonomous, driver-controlled, and end game periods. The total scores are also graphed.

The effects of changes to the robot can be traced back. For example, in between League Meet 3 (LM3) and the League Qualifier, we decided to add the IR sensor to the intake of the robot and have the LED panel signal the state of the intake. This drastically increased the average Driver-controlled period score of the team from 49 points to 90 points, a rise of 41 points. After the League Qualifier, the team realized

that the autonomous score was stagnant. By our team switching to Pedro Pathing, an increase in the average autonomous score of 66 to 95 can be observed. Noticeable increases in robot competition performance after identifying and addressing areas of improvement for the robot show the effectiveness of the co-design and co-validation strategies employed by the team and presented in the paper.

## 5. Summary

This paper explores the implementation of software-hardware co-design and co-verification methodologies to enhance the performance of a robotics system, specifically the FIRST Tech Challenge team, RevAmped Robotics. The team integrated various sensors and advanced algorithms to improve both user-controlled and autonomous functions of their robot. Key innovations include the use of parallel processing for real time state communication via LEDs, self-adaptive localization techniques, and automated testing systems for rapid error identification and correction. The paper highlights the benefits of these methodologies in achieving precise control, improved usability, and optimized performance, demonstrating the importance of integrated design and verification processes in advancing the capabilities of embedded systems in robotics.

## Acknowledgments

The authors would like to thank paper reviewers, Nancy Hou McCormack and Justin Wolf for their valuable assistance and feedback on software quality. Furthermore, the authors would like to thank the RevAmped Robotics team coach Zhunquin Wang for his excellent mentorship throughout our multiple-year FTC journey. In addition, the authors would like to thank the Oregon Robotics Tournament & Outreach Program (ORTOP), whose planning and assistance are crucial for the success of FTC events in Oregon. Finally, the authors would like to thank FTC team 10158 Scott's Bots for developing the Pedro Pathing software library.

## References

- [1] "Home." FIRST. Accessed July 31, 2024. <https://www.firstinspires.org/>.
- [2] "Game Manual Part 2 – Traditional Events." Manchester: For Inspiration and Recognition of Science and Technology, October 11, 2023.
- [3] Aha! "IT SWOT Analysis (Free Templates): AHA! Software." IT SWOT Analysis (Free templates) | Aha! software, March 11, 2024. <https://www.aha.io/roadmapping/guide/it-templates/swot>.
- [4] T Guo and M Bao. 2023. "Revamp Software Quality for Android Robots through Advanced Development and Deployment Methodologies", 41<sup>st</sup> Annual Pacific NW Software Quality Conference, Portland, OR
- [5] Bailey, Brian. "Software-Hardware Co-Design Becomes Real." Software-Hardware Co-Design Becomes Real, November 6, 2021. <https://semiengineering.com/software-hardware-co-design-becomes-real/>.
- [6] R. Mukherjee and M Purandare. 2017. "Formal Techniques for Effective Co-verification of Hardware/Software Co-designs", 54th Annual Design Automation Conference, Article No.: 35, Pages 1-6
- [7] Pedro Pathing. Accessed August 14, 2024. <https://pedropathing.com/Pedro+Pathing>.
- [8] E Kou and A Haggemiller. 2023. "Extended Kalman Filter Estimation for Autonomous Competition Robots", *Journal of Student Research*, Vol. 12 No. 1
- [9] Zeidis, K. Zimmermann. "Dynamics of a four-wheeled mobile robot with Mecanum wheels. Z Angew Math Mech" 2019; 99:e201900173. <https://doi.org/10.1002/zamm.201900173>
- [10] "Apriltag Introduction." AprilTag Introduction - FIRST Tech Challenge Docs 0.2 documentation, 2023. [https://ftc-docs.firstinspires.org/en/latest/apriltag/vision\\_portal/apriltag\\_intro.apriltag-intro.html](https://ftc-docs.firstinspires.org/en/latest/apriltag/vision_portal/apriltag_intro.apriltag-intro.html).

# We need to talk...(more) about Mental Health at Work

**Veronika Betzel**

v.betzel@reply.de

## Abstract

“Why are you so emotional?”

“Can’t you just put your private life aside?”

“We are at work, please be professional!”

Who doesn't know sentences like these?

In a world captivated by progress and achievement, mental health and well-being often find themselves relegated to the shadows or not taken seriously.

Embark on a transformative journey with me, redefining mental health discussions in the workplace. Visualize a setting where mental well-being is integral, fostering genuine support. Explore the tangible benefits for both employees and management, understanding how a supportive workplace elevates individual thriving and overall organizational success.

Offering examples of navigating mental health challenges at work and providing guidelines, insights, and ideas for supporting others, this talk goes beyond. Regardless of whether you encounter challenges or not, valuable tips and insights into supporting mental health issues in the workplace will be shared.

The question is not, “Should we talk about mental health at work?” but rather, “How can we talk about mental health at work?” This session will offer a direct answer to that question by giving examples and guidelines out of personal experiences.

Imagine a world where no one is stigmatized because of their mental health issues. Everyone is treated the same, regardless of whether they are neurodiverse, for example, or not. A world in which it makes no difference if a person is unwell physically or mentally. Just imagine if this were possible. In this paper, I will attempt to provide ideas on how we might not fully achieve this but take a step closer to this goal.

## Biography

*Veronika Betzel, also called Vero, started out studying to become a teacher, but ended up discovering a passion for Quality Assurance (QA) by chance. Initially working in QA part-time while studying, Vero made the switch to working full-time in QA after completing her bachelor's degree.*

*Vero loves the diversity of her job, but what she enjoys most is the opportunity to communicate and exchange ideas with others. She values directness and open discussions both personally and professionally. Vero loves to teach others, making her presentations both informative and engaging. Starting QA in 2017 Vero has already amassed some experience in the QA field, working for two different companies. When she's not working, Vero can be found giving back to her community as a volunteer firefighter or kicking a ball around the soccer field. Whether at work or play, Vero is a communicator at heart and always looking to make a positive impact.*

## Why is this topic so important?

Mental Health in the workplace is important whether the person is personally involved or not. This part of the paper will provide reasons why. It is divided into general and personal reasons.

### General Reasons

One of the main reasons why this topic is so important is how many people are affected by mental health issues. One in eight people worldwide lives with a mental health issue (Siemens-Energy 2023). Expressed as a percentage, that equates to 12.5%, meaning every eighth person reading this paper statistically has a mental health issue. What I want to emphasize with this is the high prevalence.

Another reason relates to a question posed by a German company called Shitshow<sup>1</sup>, an agency focused on mental health topics. For example, they offer various workshops and train people to become "Mental Health Ambassadors". In different discussions, they asked around 600 people "Would you talk openly about psychological stress at work?" Considering that at least one in eight people has mental health issues, only 8% answered, "I'm open to it". Eighteen percent said, "With HR/Leader", 37% replied, "With colleagues I trust", 10% said, "Only if there will be a change" and 27% responded, "No way! / I'm not sure". Thus, 27% are not open to discussing psychological stress at work (Shitshow 2023).

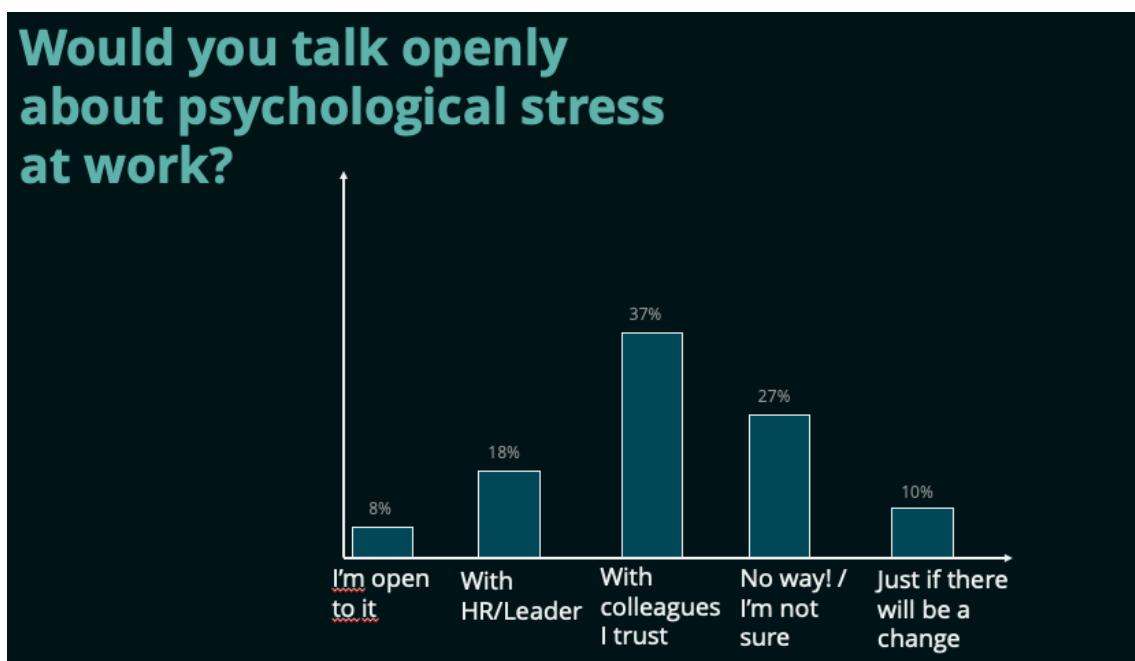


Figure 1: Shitshow 2023

It is clear that speaking about and addressing problems openly is always helpful, but so few people are willing to do so (Patterson 2002), and this is exactly what I want to emphasize in this explanation.

A third reason: There's a word for it — sanism. Sanism is 'a form of discrimination and oppression against people based on presumed mental type (e.g., ADHD or schizophrenia), mental action (e.g., stuttering), supposed intelligence, or neurology (e.g., neurotypical or autism spectrum disorder), especially those diagnosed with a mental disorder or mental illness' (Shitshow 2023). Similar to other forms of discrimination like racism or sexism, sanism leads to marginalization and systemic barriers. This term

<sup>1</sup> For more information: <https://shitshow.de/en/>

emphasizes how societal attitudes and institutions may unjustly discriminate against those with mental health challenges (Perlin 1992).

This happens because people tend to believe that there is a 'normal' type of person with a specific brain structure. Anything that deviates from this norm is considered abnormal. People tend to think in stereotypes, so when someone is placed into this specific compartment, the concept of 'sanism' is automatically assumed. This phenomenon can affect the workplace in various ways, such as employers not hiring people who fit this stereotype, giving them fewer opportunities for advancement, or simply not talking about it (Telwatte 2017).

## Personal Reasons

In the previous paragraph, general reasons were presented to explain why it's important to discuss mental health at work or why to even start doing so. In this paragraph, I want to clarify things further by providing a personal reason. I want to share my challenges, my personal 'disabilities,' and how they can negatively affect my daily work. This will be the most personal section of this entire paper. I think it's important to highlight this in the paper because I want to encourage people to talk openly about their mental health situation, and it wouldn't be a good start if I didn't begin by doing so myself. Also, it seems that I am a really good fit for the concept of "Sanism".



Figure 2: ADHD Brain<sup>2</sup>

This is my brain, or at least it shows how my brain works every second of the day when I am not sleeping. This is my ADHD. Imagine a brain working like a train that never stops and is always on the run. It would just stop to think if your physical exhaustion is so high that you can just concentrate on this and on nothing more. To mention a few side effects that it can have on my workplace — and also my private life — as shown in the picture, I have many thoughts at once, just hopping into my brain. Imagine a browser with 300 tabs open, and at least 3 of them are playing music. It can also lead to impulsive actions and being unfocused — because of the 300 tabs and the music. Additionally, others sometimes feel that I am impatient, although it is just another tab opening in my brain.

These are just my personal types of mental health issues. I am quite sure I am not the only person on the planet with ADHD or other squad members like this. Someone else could have other, stronger, or different symptoms. Now, here is the problem: A normal full-time workday is around eight hours long.

<sup>2</sup> <https://www.adhdevidence.org/blog/eight-pictures-describe-brain-mechanisms-in-adhd>

Eight hours of work. What can we do with those feelings during this time? How can we move from the assumption "Just put your feelings aside at work" to the idea of "Using them to make your work better"?

## Fundamental Assumption

The first part of this paper discussed the general importance of this topic. This part will now introduce a fundamental assumption necessary for providing guidelines to address the topic "talking about mental health at work." The fundamental assumption is that it is a three-dimensional approach to make further steps on this topic at work. The different parts will provide ideas on how this can possibly be done, communicated, and achieved. The first part is about me: "How can I improve my workspace so that it better fits me?" The second part concerns the colleagues: "What can colleagues do to build a supportive workplace culture?" The last part is about the employer: "How can the employer offer support?"

My assumption is that all three of these fundamentals — imagine a building, each of these is its own pillar — are needed to build the house of well-being at work. If one of the pillars is missing, this achievement cannot be accomplished.

## Ideas for myself

What can I do for myself to improve my individual working day?"

The first idea is very simple: It is called self-organization. Self-organization can lead to better mental health at work by reducing stress, increasing productivity, enhancing time management, and better allocating time for leisure, especially if you work in an environment with flexible working hours (Allen 2001). At first, it may sound very abstract, but I will provide an example. I found that the perfect working hours for me are primarily in the morning and then spread out. I cannot work for 6 hours straight without taking a break every one or two hours. Therefore, I structure my workday extensively in the calendar with blocks for breaks. I normally start working at 6 am after waking up at 5:30 am because this time really suits me well. I do a little stretching beforehand—this has been found to prevent my back pain. For example, I will work on Jira tickets for a customer for about two hours at the start of the day. At around 8 am, I definitely need a small coffee break for about 15 minutes. After this, I continue to work for about one more hour. Then another small break with breakfast. After this, around 10 am, my day usually continues with a daily meeting and various other meetings. Sometimes there is some time left for working on internal matters; sometimes, I go straight from meetings into my lunch break. Here, I try to take at least a one-hour break and do something like sports or go for a walk to disconnect from work. Then I continue to work for the rest of the day, again with small breaks. I try to finish my work around 3:30 pm or 4 pm, so I can still do something else other than work during the day. And that's it! This is how I structure my day, and it helps me greatly in organizing myself and reducing some internal stress. Of course, this is not the "one and only way" a workday can be planned, but it is the "one and only way" I have discovered over the years that fits me perfectly. I would encourage you to try out some planning and see what best fits you and your daily life. However, don't be too rigid and always leave some space for spontaneous activities.

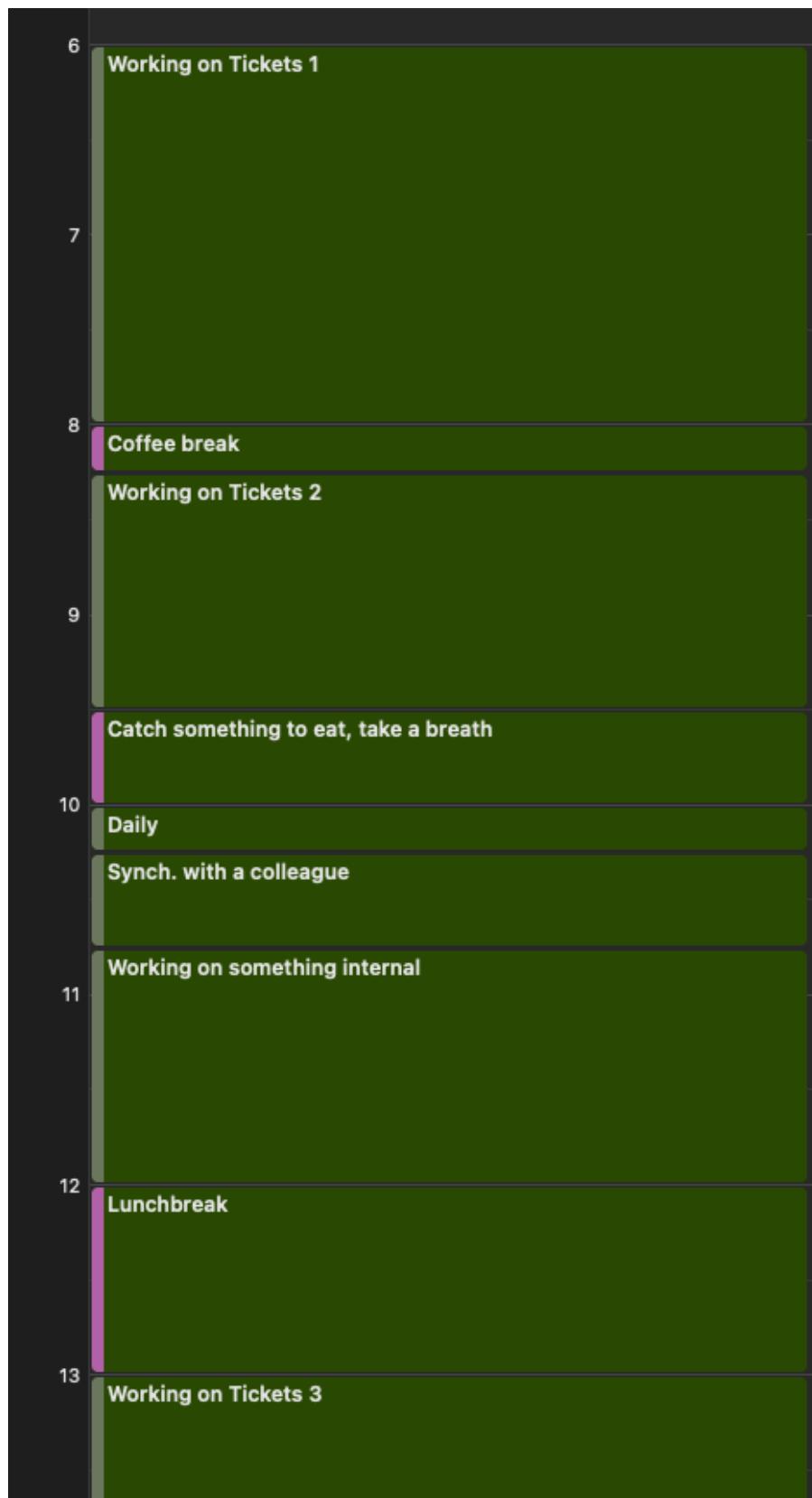


Figure 3: Typical workday example for myself

The second idea is quite similar to the first and is so important that it needs to be emphasized. The second idea would be "Listen to yourself." Ask yourself questions like, "What do you need?" and "What are your feelings?" Be aware that it's important not only to ask yourself these questions but also not to ignore the answers. Make sure to take breaks if needed, and don't forget the following statement: Being sick because of an urgent mental health issue is just the same as being sick from something visible to others.

Idea number three is not just an idea initially, but a definition. I call it Burnout vs. Boreout. What exactly does this mean? Burnout is a state of extreme exhaustion due to chronic work-related stress. It is caused by prolonged overload, excessive job demands, and a lack of adequate recovery. Symptoms could include emotional exhaustion, depersonalization, and a sense of reduced personal accomplishment (Maslach and Leiter 2016). Boreout refers to the feeling of boredom and underutilization in the workplace. It is caused by the feeling that one's skills are not adequately utilized, or tasks are too unchallenging. Possible symptoms are boredom, disinterest in work, lack of energy, and frustration (Rothlin and Werder 2008). What both have in common is that they can have some negative impact not only at the workplace but also in private life. Just to mention a few: both can lead to a negative impact on well-being, reduced job satisfaction, impaired job performance, potential for physical symptoms, an impact on relationships, workplace dissatisfaction, and an increased risk for other mental health issues (Maslach and Leiter 2016; Rothlin and Werder 2008). Why is this pointed out as an idea? In my humble opinion, everyone needs to work between these two extreme states. Thus, the employee has to find the space that is neither connected to burnout nor boreout. For myself, I have found that this space is really small and not easy to find. I get bored very easily and quickly experience the feelings and symptoms of boreout. I also get stressed easily but cannot work constructively without a bit of stress. However, if I have too much stress, I also easily veer towards burnout. In this case, I would say we go back to idea number two or forward again to idea number four.

It is necessary to listen to yourself also in a work-related context. Ask yourself questions like "What do you need?" or "What are your current feelings?" and don't ignore them.

## Ideas for my colleagues

These were just a few ideas of what can be done by a person as an individual, but of course, many of the mentioned things are also based on social interactions. And based on social interaction, there are always at least two people involved, so the next question would be: How can others help to build a supportive workplace culture?

The first idea for this is the topic of supportive communication. Communication is always a two-sided thing because you have the person who is communicating and the other one who is listening or recognizing (Berlo 1960). It will not work if just one person is communicating "correctly." So, what are possible strategies? Possible strategies could include Active Listening, Setting Boundaries, Maintaining a Non-judgmental Attitude, and Showing Empathy. By Active Listening, I mean something like giving the speaker your full attention (by nodding, for example, if you appreciate what is said).

Another point would be to avoid interrupting. Setting boundaries means being clear about your own limits but also respecting those of others. For example, if I need to prepare a talk or write a paper, I require deep focus and should not be interrupted. Some music is nice and necessary, but perhaps not messages. Therefore, I set a blocker on my text messaging tool at work for this deep focus period, setting a boundary that others need to respect.

Having a non-judgmental attitude means avoiding making judgments. If a person is doing something a certain way, there is always a reason behind it. Imagine a situation where a meeting is set up at 10 am. One person is always late to this meeting. A first reaction might be, "Oh, this person is just too irresponsible to make it on time." What could you do instead? You could communicate with the person

and ask why they are always late. Maybe the time isn't suitable, perhaps there is a meeting beforehand, or maybe a reminder is missing, but asking will handle the situation better than simply judging.

By empathy, I mean the idea of understanding and acknowledging the emotions of others. Emotions are basic for human beings; some people are more emotional than others—especially during work. From my perspective, I get called “emotional” a lot. I tend to think with my brain but also with my heart; however, I don't like being called “emotional.” If I am emotional, it is more helpful to ask me why that is the case. Most of the time, there is a reason behind it, like being stressed or other valid reasons.

One different idea for the part “What can I do as a colleague?” is the concept of creating a safe space at work. A safe space has numerous positive impacts on employees and overall organizational benefit, such as encouraging open communication, fostering psychological safety, and other great benefits (Edmondson 2019). For me personally, this means having a space where I can be emotional without being judged, a place where people can ask me, or I can ask if everything is okay, if there is a feeling that something is wrong; a place where no one needs to answer because a boundary was set there; a place without discrimination and that supports creativity. For me, it would also mean that I can talk openly about having a girlfriend without facing any problems or sideways glances about the issue.

Of course, this cannot be done only by myself or together with my colleagues. To tackle this topic, a third party is needed.

## Ideas for my employer

So, after gathering some ideas and guidelines for myself and other colleagues, the last part addresses the employer. How can the employer support?

First of all, I want to point out that in my opinion, it is not necessary to involve an external party to initiate discussions on “Mental Health,” and making a start is definitely better than doing nothing. A start could also involve making people aware of the issue by raising the topic—for example, by holding a talk about it. This can be done by just one person.

The employer can also support by providing emergency plans and designating at least one person (or more) who is responsible for them. An idea in this case would be to establish some kind of “Mental Health Ambassador,” for example. In my current company, we have an external company that provides urgent help if needed. You can simply write them an email or call them, and you get help from professionals as soon as possible. It is not just for work-related problems; they help deal with serious life situations where assistance is needed.

Another idea concerns company values. Don't just talk about them, write them down. Make sure everyone understands them and aligns with them. In my current company, we have a Confluence page where the company values are written down. When we are searching for a new employee, we focus not only on technical skills but also on personal and social skills. Maybe we go to dinner together to get to know each other better. We conduct not just one interview but also several more, including possible new team members. This way, during decision-making, it can be seen whether the person respects the company values and is not just a technical fit but also a personal fit for the company.

Support work-life balance by looking out for the employers, perhaps supporting sports programs, after-work events, respecting free time, and so on. In my current company, we have a sports program called “Hansefit.” This means that the employee can go to several gyms, sports clubs, or other sportive activities without having to pay. Furthermore, we regularly hold socializing events that help at least me to maintain a balance between doing “real” work and socializing.

Last but not least, in my humble opinion, it is necessary to evaluate the status quo. Don't adopt the attitude as an employer that “If I don't know anything about problems regarding mental health, then there

are no problems, so it's not my problem." Ask the employees about it. Try to define a status quo to have a chance to change it if necessary. As a side note, there are also some ready-to-use surveys for evaluation.

## Conclusion

I have just talked to you about mental at work. My goal was to give ideas to help you create a more supportive workplace culture and discuss potential guidelines for you. It shouldn't matter if you are an employer, colleague or if you are taking inventory of your own mental well-being at the workplace. Mental Health at work should be everyone's concern...let's talk about it!

Right from the beginning, I mentioned three parts necessary to achieve what seems to be "well-being at work." So, to summarize, well-being at work cannot be achieved alone, nor by my colleagues or even the employer alone. For me, it requires three pillars to build the house of well-being at work that influences it. First, the employer, who creates the space for it and provides opportunities by defining, evaluating, and asking. Second, it needs the colleagues who collaborate with each other and take the opportunities provided by the employer, respecting the framework that was created by them. And last but not least, it needs myself, who works at a company, collaborates with colleagues, is open to communicating problems openly, and also has space to support others.



*Figure 4: The house of well-being at work*

To finalize this, I would like to end with a quote found in the World Wide Web: "8 billion people need to redefine what success looks like. It's not to make a billion dollars, it's to wake up in the morning and actually be in a good mood." (Vaynerchuk 2023)

## References

- Allen, David. 2001. *Getting Things Done: The Art of Stress-Free Productivity*. New York: Penguin Books.
- Berlo, David K. 1960. *The Process of Communication: An Introduction to Theory and Practice*. New York: Holt, Rinehart, and Winston.
- Edmondson, Amy C. 2019. *The Fearless Organization: Creating Psychological Safety in the Workplace for Learning, Innovation, and Growth*. Hoboken, NJ: Wiley.
- Maslach, Christina, and Michael P. Leiter. 2016. "Understanding the burnout experience: Recent research and its implications for psychiatry." *World Psychiatry* 15, no. 2 (2016): 103-111.
- Patterson, Kerry and others. 2022. *Crucial Conversations: Tools for Talking When Stakes are High*. New-York: McGraw-Hill
- Perlin, Michael L. 1992. "Psychiatry and the Law: The Impact of Sanism." *American Journal of Forensic Psychiatry* 13, no. 1: 7-36.
- Rothlin, Philippe, and Peter R. Werder. 2008. *Boreout!: Overcoming Workplace Demotivation*. London: Kogan Page.
- Shitshow – Mental Health Consultancy. 2023. "Sanism", Linked-In, posted May 2023, [https://www.linkedin.com/posts/shitshow\\_sanism-sanismus-ableismus-activity-7026594952197365760-IGhl?utm\\_source=share&utm\\_medium=member\\_desktop](https://www.linkedin.com/posts/shitshow_sanism-sanismus-ableismus-activity-7026594952197365760-IGhl?utm_source=share&utm_medium=member_desktop) (accessed May 7, 2024)
- Shitshow – Mental Health Consultancy. 2023. "Würdest du auf der Arbeit über deine psychischen Belastungen sprechen?", Linked-In, posted May 2023, [https://www.linkedin.com/posts/shitshow\\_neues-ausbildungsprogramm-unser-mental-health-activity-6998256195564150784-9XEj/?utm\\_source=share&utm\\_medium=member\\_desktop](https://www.linkedin.com/posts/shitshow_neues-ausbildungsprogramm-unser-mental-health-activity-6998256195564150784-9XEj/?utm_source=share&utm_medium=member_desktop) (accessed May 7, 2024)
- Siemens Energy. 2023. "Breaking the silence on mental health" Siemens Energy, posted March 27, [https://www.siemens-energy.com/global/en/home/stories/breaking-the-silence-on-mental-health.html?gclid=Cj0KCQjwuZGnBhD1ARIsACxbAVizlyvAvnYdhopUOrxSV\\_mXfbSgzPKnETn\\_sOP9ylATi90IQjwBZwaArBnEALw\\_wCB](https://www.siemens-energy.com/global/en/home/stories/breaking-the-silence-on-mental-health.html?gclid=Cj0KCQjwuZGnBhD1ARIsACxbAVizlyvAvnYdhopUOrxSV_mXfbSgzPKnETn_sOP9ylATi90IQjwBZwaArBnEALw_wCB) (accessed May 7, 2024)
- Telwatte, Aspara and others. 2017. *Workplace accommodations for employees with disabilities: A multilevel model of employer decision-making*. Rehabilitation psychology.
- The ADHD Evidence Project. 2021, posted May 2021, <https://www.adhdevidence.org/blog/eight-pictures-describe-brain-mechanisms-in-adhd> (accessed May 21, 2024)
- Vaynerchuk, Gary. 2023, Linked-In, posted 2023, [https://www.linkedin.com/posts/garyvaynerchuk\\_7-billion-people-need-to-change-the-conversation-activity-6947278918622093312-k523/](https://www.linkedin.com/posts/garyvaynerchuk_7-billion-people-need-to-change-the-conversation-activity-6947278918622093312-k523/) (accessed May 7, 2024)

# QA Authority: From Zero to Hero

Anton Bolotin

[ambolotin@gmail.com](mailto:ambolotin@gmail.com)

## Abstract

In the modern world some people hold the opinion that Quality Assurance Specialists are not 'real' IT professionals, and could be carried out by anyone, without any specific training or experience. These views are often based on a misunderstanding of what QA actually does and the value that QA brings to projects.

In my presentation, I will explore the reasons for such misunderstandings and present strategies that will help QA specialists increase their authority and influence within the team and project. We will discuss methods of effective communication with other team members, demonstrating the value of your work, and ways to participate in development processes to become an indispensable team member.

## Biography

Anton Bolotin has over 7 years of experience as a QA and Scrum Master. He has extensive experience with manual testing, automation, load testing, and security. He has implemented new features in testing projects, built processes from scratch, and managed teams of up to 8 people. He currently works as a QA specialist and mentor, and he also manages his own projects.

# Introduction

Have you ever experienced toxicity or lack of respect from coworkers because you work as a QA? Well, in my experience I can tell you that unfortunately this happens quite often.

We were all once at the beginning of our careers and were excited to see the world of IT, where specialists of different profiles come together in teams and create new amazing products. Product managers, designers, developers, analysts and of course QA specialists.

A good team and colleagues are often an important factor that can keep employees from leaving a company if other factors, such as salary, do not satisfy the employee.

I have been working in QA for a long time and giving personal consultations for a while now. During this time, I have talked to a great number of people: those who just wanted to step on the path of a quality assurance specialist, those who were already actively working in this profession, and those who for various reasons stopped working as QA. And, of course, I asked them about the processes in the company, relationships with colleagues, and in general about their experience of working in different teams.

Here are a few situations and statements that many QA professionals I know have faced

- “Testing is the easiest field in IT, anyone can get in”
- “Testers are not programmers; they don't write code.”
- “Testers don't know anything except how to push buttons and make regressions testing.”
- “Monkey tester”
- “Automation tests are fake; they are not useful.”
- “QA's opinion is not counted in team meetings.”
- “QA has no authority in the team to push their ideas.”

In this paper, we will look at some of the challenges that QA professionals face that downplay the importance of their work, and ten key things that they can do to showcase the value they bring to their teams.

## What to do?

To understand how to fix a problem, you must first understand what caused it.

### Let's take a look at:

- what is the reason for such attitude
- what can be done about it
- how to gain the respect of your teammates

I decided to explore the problem a little deeper and look at it from a different perspective. To do this, I have talked to developers, analysts and managers of various companies about their experience of working and interacting with testers.

Most of them evaluated the experience of interaction with testers in their teams as positive. However there was also a place for negative experience of working with some QA specialists. What could be the reason for this?

As a result of the research I made the following conclusions

## Reasons

- Your colleagues don't see the results of your work.
- That doesn't mean you're not doing a good job
- It means you need visualization and transparency.
- Achievements of programmers are very visible - a feature implemented, a bug fixed, a new toolkit that speeds up the work of the team.
- QA has some different working artifacts that are not always visible to others.

Excerpt from PNSQC Proceedings

PNSQC.ORG

Copies may not be made or distributed for commercial use

## Solution

Here are ten key things that QA professionals can do, to help raise the profile of their work within their teams, and to showcase and demonstrate the value that they bring to their projects.

### 1 Ask your teammates to help you with QA process improvement.

Taking care of product quality is not something that should rest solely on the shoulders of QA specialists. While QA teams are responsible for identifying bugs and ensuring product quality, quality assurance is ultimately a team effort. Every member of the team has a role to play in delivering a high-quality product, and involving others in the improvement of QA processes not only enhances the end result but also increases collaboration and mutual understanding among team members. When everyone shares the responsibility for quality, it becomes easier to achieve the desired outcome: a stable, well-functioning product.

Creating opportunities for teammates to contribute to quality improvements helps build a sense of ownership. When developers, analysts, and other stakeholders are involved, they see firsthand how their roles directly impact quality. This engagement shifts the perception of QA from being seen as a bottleneck or afterthought to being viewed as an integral part of product development.

- **Analysts:** Ask analysts to review your test cases. Their perspective can bring clarity to test coverage and help identify areas that need more thorough testing. Analysts work closely with business requirements, so their feedback ensures that the QA process aligns with end-user expectations. This will ensure that tests focus on the most critical business logic and cover edge cases that may not be obvious at first glance.
- **Backend developers:** Engage backend developers to create or modify tools that help generate test data more efficiently. By collaborating with them, you can ensure that complex datasets or difficult-to-manage test environments are automated or simplified. For example, they can develop scripts that generate predefined data sets or mock services that mirror real production behavior. This not only speeds up the testing process but also reduces the risk of inconsistencies and human error when preparing data manually.
- **Frontend developers:** Collaborate with frontend developers to add test IDs to UI elements, making it easier to write and maintain automated tests. Without proper identifiers, tests can become fragile and hard to manage as the UI evolves. By incorporating test-friendly code (such as unique IDs or data attributes), frontend developers contribute directly to making automated testing more efficient, reliable, and easier to debug.

As a result, it will help your colleagues understand the complexity and challenges QA teams face, increase empathy and collaboration, and help make your performance more transparent to your colleagues.

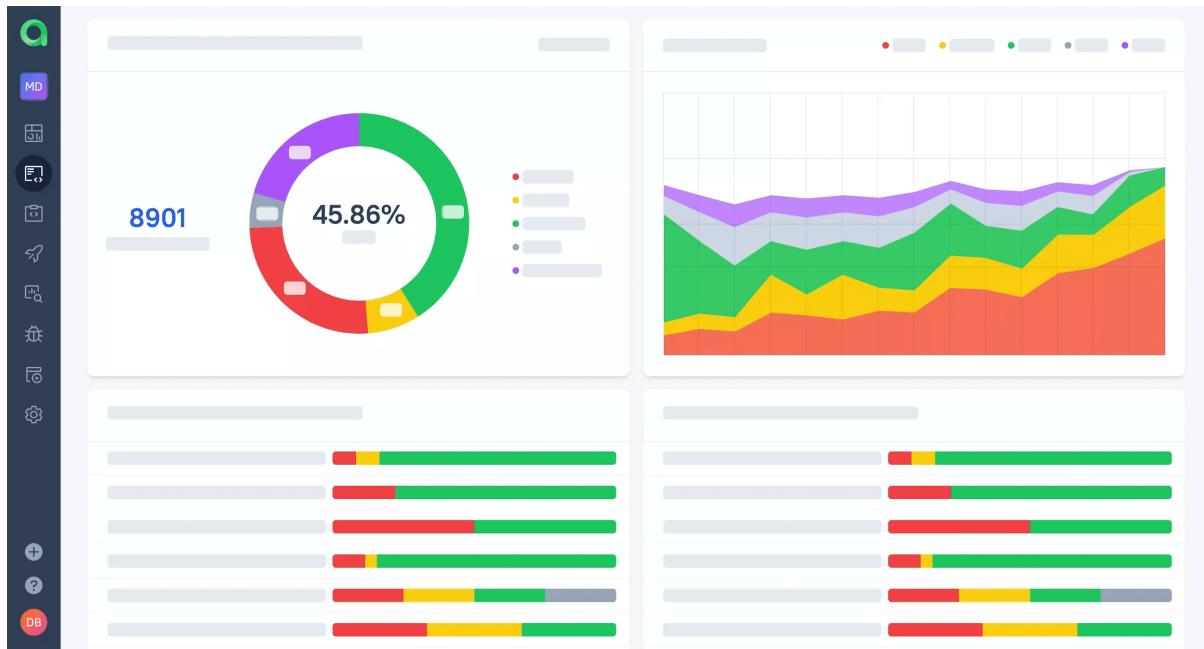
## **2 Visualize the results of your work. Dashboards**

One of the most effective ways to demonstrate the value of QA work and increase transparency within the team is through the use of visual dashboards. Dashboards provide a clear, real-time view of the

current state of testing and quality assurance efforts, allowing both QA teams and other stakeholders to quickly understand progress and identify potential issues.

Built-in tools in test management systems make it easy to create these dashboards. You can track important metrics such as the number of active defects, test pass rate, test case execution progress, and the number of days left until the release.

Creating a dashboard that is regularly updated and accessible to your colleagues allows them to follow the progress in real time. Stakeholders from product management to development will appreciate the transparency and the ability to assess risks and plan their tasks more effectively based on up-to-date testing data



### 3 Test Automation

#### ***Start writing automation tests.***

Starting with test automation can be a game-changer for your role as a QA specialist. The moment you begin writing automated tests, you elevate your status within the team and position yourself as a more versatile expert, as someone who is not only testing but also contributing directly to the development process through coding.

The secret is that it is not as hard as it seems at first sight. There are many test automation frameworks that work on the principle of Record and Play, which are ideal for beginners or those without extensive coding skills. For examples Selenium IDE, Testim, and Katalon Studio. These tools allow you to create automated tests through a simple point-and-click interface, making them accessible to a wide range of professionals.

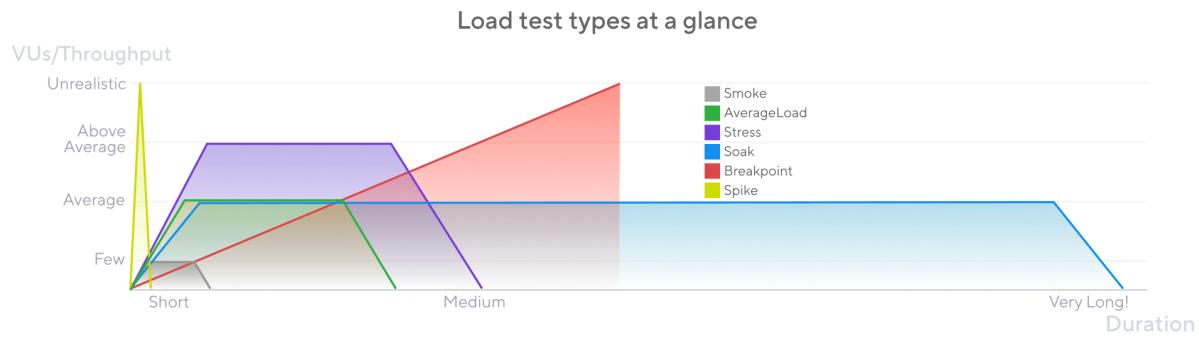
For QA specialists with coding experience, more advanced frameworks like Playwright, Selenium WebDriver and Cypress offer greater flexibility and control. These tools enable the creation of complex, highly customized tests, often required in larger or more sophisticated testing environments.

And the best way is teaming up with developers to think about what tool, language and framework to implement. Often it's beneficial to choose the same language as the developers are using, as they can then help troubleshoot any issues you might have, and they can also step in and write test cases.

And remember, you can always ask ChatGPT to help you navigate through any challenges you encounter while setting up or using these frameworks.

## 4 Performance Testing

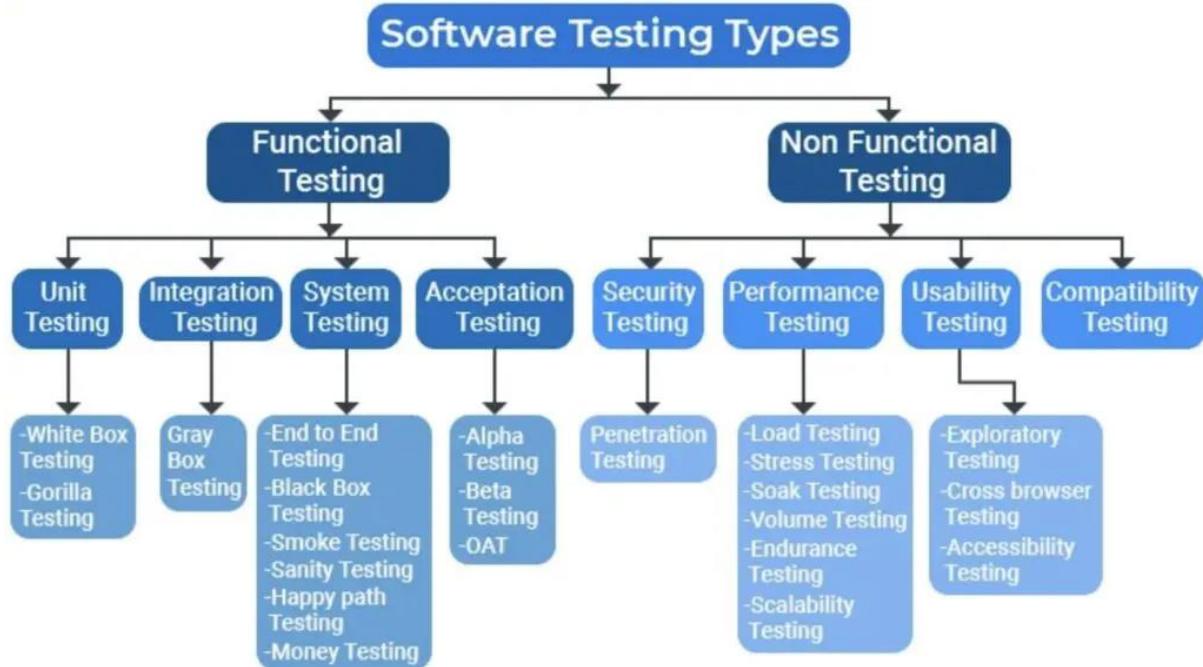
If you have already started doing test automation, you can also try to initiate load testing on the project. After all, a tester who does not only manual testing, but also automation and load testing, definitely earns respect.



One of the most popular tools for load testing is Apache jMeter. It's a powerful tool that allows you to simulate multiple users interacting with your application to assess its performance under load. To get started, you might use Official jMeter Documentation: A great place to start, offering comprehensive guides on installation, setup, and basic test creation. But that's not the easiest way, so let's look at some other options.

## 5 Other types of testing

Start other types of testing on the project. For example, Security testing: you can introduce security checklist checks at each release or testing of documentation, design, cross-browser testing, and accessibility testing.



Accessibility testing is becoming increasingly important to ensure that your application is usable by people with disabilities. There are many freely available tools to help you get started with accessibility testing:

- Axe byDeque: A powerful and easy-to-use browser extension that automatically identifies accessibility issues on web pages.
- WAVE by WebAIM: Another browser extension that provides visual feedback about the accessibility of your web content.
- Lighthouse: A built-in tool in Google Chrome's DevTools that includes an accessibility audit feature.

Incorporating these types of testing into your project not only broadens your testing coverage but also helps in building a more robust and user-friendly product.

## 6 New tools

Introducing new tools to your team can greatly improve efficiency and enhance your role as a QA expert. By selecting the right tool for a specific need, you not only optimize your own workflow but also offer benefits to the entire team. One powerful way to make an impact is by implementing a Test Management System for writing and storing test cases, moving away from less structured options like

Excerpt from PNSQC Proceedings

PNSQC.ORG

Copies may not be made or distributed for commercial use

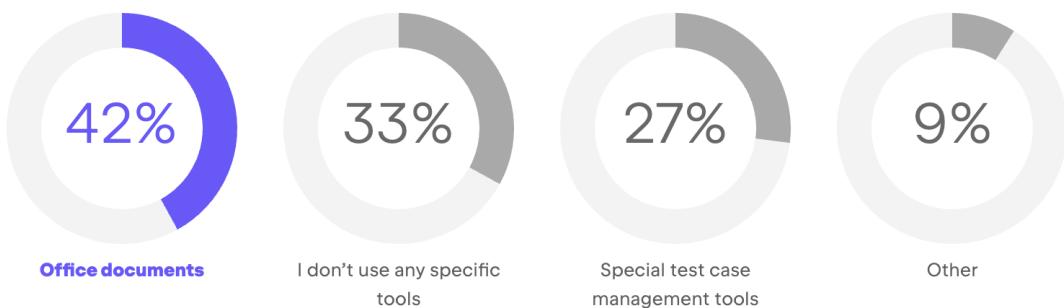
Google Docs or spreadsheets. A dedicated tool will help you organize test cases, track execution, and report results more effectively, ensuring nothing falls through the cracks.

If your team already uses a task tracker like Jira, there are many plugins and integrations that can be added from its library. For example, plugins like Xray or Zephyr can help manage test cases within Jira, linking them directly to user stories and issues. This allows for better tracking of test coverage, defect trends, and overall project health.

Another innovative tool to consider is GitHub Copilot, an AI-based code completion tool that helps automate repetitive tasks in code writing. If your work involves writing automation scripts, Copilot can significantly speed up the process by suggesting code snippets, helping you debug, and providing quick solutions to common problems. Sharing these capabilities with your team will likely spark interest and curiosity, as many of your colleagues may want to leverage the tool for their own coding tasks.

Implementing new tools not only improves your work but also positions you as a forward-thinking professional who is committed to continuous improvement. Sharing your experience with these tools will encourage your teammates to embrace new technologies and support a culture of innovation within the team.

#### **What tools do you use to store test cases?**



*The State of Developer Ecosystem 2023 - Testing*

Source: <https://www.jetbrains.com/lp/devcosystem-2023/>

## **7 Create your own meetings and showcases**

If you find that existing meetings or calls are not allowing you to identify how to contribute, why not take the initiative and organize your own meetings dedicated to QA topics? This can be a great way to share your expertise, engage your colleagues, and increase QA visibility within the team. Here are some ways to do this effectively:

- **Plan a meeting on quarterly testing goals:** Set up a session to discuss the testing strategy and objectives for the upcoming quarter. This will give everyone a clear understanding of how QA efforts align with broader business goals and project timelines. It also allows you to collaborate with your colleagues on prioritizing features, test coverage, and potential risks.
- **Conduct training sessions:** Host a workshop or training session for your colleagues on the fundamentals of QA or on using specific QA tools. This can be particularly helpful for non-QA team members, such as developers or product managers, who may not be familiar with the intricacies of testing. Training can also cover topics like automated testing, test case management, or defect reporting.
- **Organize a testing showcase:** Hold a quarterly or bi-annual meeting where you present key testing achievements, such as defects discovered, challenging scenarios, or important metrics that show progress and areas for improvement. By showcasing your work, you highlight the importance of QA and provide the team with a clear understanding of the value it brings to the project.

Taking the initiative to organize these meetings not only showcases your leadership and communication skills, but it also helps to create a more collaborative, transparent environment. I once saw a situation where the morale of an entire department was low, so I organized a department retro in the form of a game. The result was not only a great boost to the team's energy but also an unexpected improvement in my performance review score. This goes to show that even informal, creative meetings can have a significant impact.

## 8 Conferences and meetups

Excerpt from PNSQC Proceedings

Copies may not be made or distributed for commercial use

PNSQC.ORG

Attending conferences and meetups is an excellent way to stay updated on industry trends, connect with professionals, and expand your knowledge base. There are two main ways you can leverage these events to benefit both yourself and your team

Easy way: Go to a conference or meetup as a listener and absorb new ideas, insights, and best practices. After attending, prepare a presentation or informal discussion to share what you learned with your colleagues. This allows your team to benefit from your experience, even if they weren't able to attend the event themselves.

Cool way: For an even bigger impact, consider attending a conference or meetup as a speaker. Presenting your own experiences, challenges, or success stories not only raises your professional profile but also positions you as an expert in your field. Afterward, share your speaking experience with your colleagues.

Whether as a listener or a speaker, engaging with the QA community through conferences and meetups enhances your skill set and can motivate your colleagues to take similar steps toward professional development.

## **9 Became Scrum master**

A story from personal experience. At one of my jobs, we worked in Scrum. And we had a special role of Scrum Master, which could be combined by any member of the team. So, I became a Scrum Master in the team, combining the main role of QA. And somehow, I got more authority in the team, becoming the second after the team leader and having a huge opportunity to influence the team's decisions and changes in the processes. A little bit off the Scrum way, but just know that there is such a way too and it works. And what to do with authority and the ability to influence - everyone must decide for themselves.

If you're interested in learning how to become a Scrum Master, here are some starting points that might be helpful:

- Scrum.org offers the Professional Scrum Master (PSM) certification, which includes a variety of training materials and practice assessments.
- Coursera has courses like "Introduction to Agile Development and Scrum" by IBM, which provides a solid foundation in Scrum practices.
- Scrum Alliance provides a well-recognized Certified ScrumMaster (CSM) course that combines theoretical knowledge with practical exercises.

These resources can help you gain the necessary knowledge and skills to effectively step into the role of a Scrum Master and leverage your influence within the team.

## **10 Be yourself**

Throughout this paper, I have offered various advice on how to demonstrate your value as a QA professional and gain authority among your colleagues. The goal was to highlight how many opportunities in QA that you can explore if you are passionate about it.

Feel free to take and apply any of these approaches if they fit your situation. However, it is important to remember that none of this advice is mandatory. You are already a professional with your own unique skills and contributions.

After all, the best approach is to simply be yourself.

## **Conclusion**

In the journey of working as a QA specialist, you may encounter challenges related to how others perceive your role. However, by implementing the strategies discussed you can build stronger relationships and influence within your team.

I hope that my advice helps you navigate your journey in QA with greater confidence and effectiveness.

# The Future is Finding Zero Bugs (in QA)

**Leslie Brooks (aka The BDD Coach)**

Leslie@TheBDDCoach.com

## Abstract

Depending on QA to catch bugs is inefficient – it is far cheaper if we never create the bugs – and it is far cheaper if we catch them before the Developers open a pull request. Depending on QA to catch bugs also means that QA takes the hit if requirements are late and the Developers take too long. The date doesn't move; QA gets crunched. Shifting all the way left - so far left that we prevent bugs from ever being written, and so far left that we deliver automated tests to the developers before they finish coding, eliminates these problems. BUT, the only way I know to do that is to use Specification by Example/BDD, and that is a methodology change that requires training and coaching, time, and buy-in from the Product Owners, Developers, and QA. I have tried it without that buy-in, but it doesn't work well. However, with that buy-in and the appropriate training and coaching I have seen a team deliver monthly releases for a year - and only two bugs into QA!

## Biography

*I have decades of experience in QA and specifically in automated testing, but I am always laser focused on value. Automated tests that cost too much to maintain don't deliver the value that we should demand. I love teaching Specification by Example/BDD; I love the tremendous improvements it can bring to the entire organization. When we are able to write better requirements, write better code (and fewer bugs), find and fix bugs faster, and deliver better quality code faster, then I feel an enormous sense of accomplishment.*

Copyright The BDD Coach LLC, 2024

## 1 Introduction

In the world we live in today, we expect developers to write code, and we expect that code to contain bugs. We expect the QA team to catch most of those bugs, but we expect that some of the bugs will escape into Production.

This is a world in which no one is truly happy – the Developers aren't happy because they spend too much time fixing bugs. The QA team isn't happy because they suffer from schedule cram-down – if the developers eat up too much of the schedule, the QA team is told they have less time to test. The customers aren't happy because they suffer with bad software, and the Product Owner isn't happy because the delivered software doesn't always do what they wanted it to.

There is a way to fix these problems; some companies are already doing it, but for most companies it is still in the future. This future is one where we truly 'shift left' and:

- The QA team doesn't run any tests (and therefore finds zero bugs)
- No bugs escape to Production

If we don't catch them in QA and they don't escape into Production, the only place left to prevent, catch, and fix them is in Development. In this future:

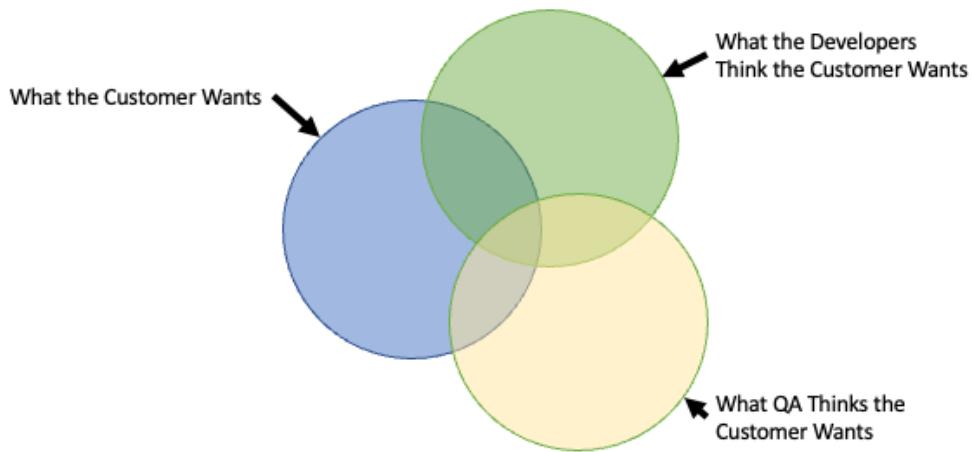
- The QA team delivers automated tests to the Developers before the Developers finish writing the code
- The Developers run the tests
- The Developers catch (and fix) all of the bugs before opening a pull request

This really is possible – there are companies doing it today.

## 2 How Does Our Requirements->Development->Test->Deliver Methodology Produce Bugs?

### 2.1 How Do Our Requirements Produce Bugs?

We write requirements so the developers will understand what the Product Owner wants to deliver to the customers. The developers read these requirements and say "OK; I understand what the customer wants, and I understand what I need to develop." The QA team reads these requirements and say "OK; I understand what we need to test" – and we wind up with this:



Of course this results in bugs. I like to talk about two kinds of bugs:

### **2.1.1 Little-'b' bugs**

Little-b bugs are the ones where the code doesn't do what the developer intended – off-by-one errors are common examples. Other Little-b bugs could be that a database field isn't large enough for some of the data that we need to store in it, or that a phone number field accepts alpha characters, etc. Little-b bugs look like this:



Little-b bugs are a direct result of incomplete requirements or incomplete test cases. If we have comprehensive requirements then we can avoid creating Little-b bugs; if we have comprehensive tests that cover all of the corner cases and boundary conditions then we can avoid delivering them into Production.

### **2.1.2 Big-'B' Bugs**

Big-'B' Bugs are the ones where the code does exactly what the developer intended; it just isn't what the customer wanted. These can be very costly – the entire team may have spent an entire sprint – hopefully not two or three sprints – writing really good, clean, high-quality code that we can now throw away, because it doesn't do what the customer wants! Big-B Bugs look like this:



Big-B Bugs are created when the developer's understanding of what the customer wants isn't the same as the Product Owner's understanding, and are a direct result of imprecise requirements.

## 2.2 How Do Our Developers Create Bugs?

Our developers don't want to create bugs – many developers are highly skilled, very careful in their coding, write really good unit tests, and use TDD (Test Driven Development). These all contribute to reducing the number of bugs that are created, but a typical developer still spends 30% to 40% of their time fixing bugs.

Developers create Little-b bugs by typing the wrong thing – entering the wrong field type when creating the database, or typing '`<=`' when they should have typed '`<`', etc. It is difficult to completely prevent these mistakes, but they can be caught if we have comprehensive tests.

Developers create Big-B Bugs by misunderstanding the requirements; we can prevent these bugs from ever being created by writing really clear requirements – and that is why Specification by Example/BDD was invented.

## 2.3 How Does Our Testing Fail to Find Bugs?

### 2.3.1 Incorrect Test Cases

If the test cases are wrong then we may run the tests but not find a bug. Test cases can be wrong because the QA team misunderstood the requirements; we can prevent this misunderstanding by writing really clear requirements. Test cases can also *become* wrong – they were correct when they were written, but the requirements have changed. When requirements change we are supposed to review the test cases that

Excerpt from PNSQC Proceedings

Copies may not be made or distributed for commercial use

PNSQC.ORG

Page 5

map to that requirement – but the mapping could be wrong, or the person who reviews the test case might not make the right changes, or the review could be missed altogether. Specification by Example/BDD will definitely fix these process problems, because the requirements **are** the test cases. The test cases can only be wrong if the requirements are wrong!

### **2.3.2 Missing Test Cases**

If we don't have all of the test cases we need – if we don't have comprehensive test cases – then bugs can also escape into Production. Specification by Example/BDD won't guarantee that these bugs are caught, but it definitely helps.

For example, if we have customer accounts with three different statuses – Active, Inactive, Suspended, etc. – then we would need to write three test cases to verify correct system behavior for each of those statuses. The Product Owner is very unlikely to review the test cases; if the test case for Suspended accounts is left out then it would simply not be tested.

Some teams would automate this with one data driven test case with all of the statuses in a spreadsheet. The Product Owner will almost certainly never review the spreadsheet, so if an account status is missed then it would simply not be tested. SBE says that the **requirement** must include the concrete examples; SBE doesn't allow it to be hidden away in a spreadsheet. Using Gherkin we would write a Scenario Outline that included all of the statuses; the PO, Developers, and QA would all review this requirement and it is very likely that someone would notice that a status was missing.

## **3 How Can We Eliminate Bugs?**

Our current methodology – even a good Agile methodology:

- Allows us to create bugs
- Assumes that we will create bugs
- Assumes that we must have a QA team to catch the bugs
- Assumes that some bugs will escape into Production and be found later

If the problem lies in the methodology, then we can only fix it by changing the methodology. We can't fix it by giving our developers more training, or hiring more QA people, or switching to a better programming language or framework, or using a new QA tool.

### **3.1 How Can We Eliminate Bugs While Writing Requirements?**

We can eliminate the Big-B Bugs by writing requirements using Specification By Example (SBE). SBE says that all requirements must include concrete examples, and must follow a clear 'Given these preconditions, When this event happens, Then this is the expected result' pattern. Both concrete examples and the clear Given, When, Then pattern help to eliminate (prevent) Big-B Bugs – it is much harder to misunderstand the requirements when they are written this way. (This Given, When, Then format is called 'Gherkin'.)

Using concrete examples also helps prevent Little-b bugs, because when we see concrete examples that leads us to think of other concrete examples. This helps us write comprehensive requirements – requirements that include all of the corner cases and boundary conditions.

### **3.2 How Can We Eliminate Bugs While Writing the Code?**

#### **3.2.1 Write Clearer Requirements**

Using Specification by Example for writing requirements helps us write clearer requirements; it is much easier to understand what the customer really wants when the requirements include concrete examples.

### 3.2.2 Eliminate Incorrect and Missing Test Cases

In the world we live in today, eliminating incorrect test cases and making certain that our test cases cover all of the requirements, is very difficult. We must carefully read the requirements, think through the test cases required, and write all of the test cases. Writing test cases:

- Is extra work
- Takes extra time (and ensures that we *cannot* deliver automation in-sprint)
- Makes it possible for the QA team to misunderstand what the customer wants
- Makes for extra work when the requirements change – now test cases must be examined and possibly updated

The only way to effectively solve these problems is to not write any test cases – if we don't have any test cases, then we don't have any of the associated problems. Of course, we would also deliver all of the bugs into Production; that wouldn't be good!

However, if we use the SBE methodology and we do a good job of writing requirements up front, we eliminate all of these problems:

- 1) Our requirements and our test cases are one and the same; we don't have requirements and separate test cases, so we don't have to waste any time writing or maintaining separate test cases.
- 2) Our requirements must describe all of the features we plan to deliver, so we have a good start on completeness.
- 3) If we are careful to list all of the boundary conditions in our requirements, then we really can have complete requirements. Thinking of all of the boundary conditions is easier when requirements must include concrete examples.
- 4) If we always change the requirements before changing the code, then our test cases (the requirements themselves) are always up to date.

This doesn't mean that the QA team and the developers shouldn't *think about* test cases – they should. However, they should think of possible boundary conditions and design problems up front, so that the requirements can be fixed before the developers begin writing code. Getting correct and complete requirements before the developers begin coding means they will write fewer bugs, which means they will spend less time fixing bugs.

### 3.2.3 Deliver Automation In-Sprint

Rather than writing test cases, SBE says that we should directly automate our requirements. That is, the automation team should write an interpreter that treats our requirements as a formal grammar and runs them against our implementation<sup>1</sup>. If we do this:

- We don't waste time writing test cases
- The QA team can't misunderstand what the customer wants – we are literally running the requirements, not running test cases that we wrote based on our interpretation of the requirements.
- There are no test cases to be examined and possibly updated when the requirements change

If the Product Owner decides to change the requirements, and can make the changes without needing new requirements grammar, then changing the requirements does not result in any automation work for the QA team. If we do a good job of writing our requirements – if we write statements that can be easily re-used – then the Product Owner will be able to change existing requirements and even write many new requirements using existing grammar. When we re-use existing grammar – grammar that our interpreter already understands and knows how to automate – then the QA team has nothing to automate.

---

<sup>1</sup> Not everything should be automated; for example some graphical tests may be very difficult to automate but very easy for a human.

We won't deliver automation in-sprint for the first sprint, or probably for the first several sprints, but once we reach the point where the PO is using mostly existing grammar for writing new requirements, then we can consistently deliver in-sprint automation. When the PO can write twenty or thirty new requirements using existing grammar, and need just one new 'When' statement and one new 'Then' statement, then the QA team only has to automate those two statements to fully automate all of the new requirements.

Looking at this another way, in Programming 101 our professors taught us to make the upper layers of our programs as abstract as possible, and to push the implementation details as far down as possible. Test cases violate this rule in the worst way – our test cases are full of references to account numbers, passwords, names of text fields and database fields, URLs, and many other implementation details. When the requirements change we must make many, many test case and code changes. If we write requirements that talk about business purpose rather than about implementation, then:

- 1) The top layer of our program (our requirements) is more abstract, so we get lots of reuse out of our grammar and our automation
- 2) Implementation details have been pushed down to lower layers  
When requirements change, very little automation code has to change – and sometimes none at all!

### **3.2.4 Run the Automation Before Opening Pull Requests**

Once we are consistently delivering automation in-sprint, the developers can run the requirements before they open a pull request. If any of the requirements fail, then there is a bug and they aren't ready to open the pull request. Once all of the requirements pass, then they can open the pull request, knowing that they haven't written or introduced any bugs. Of course the CI/CD pipeline should also run the requirements, either on every merge or nightly, just in case a developer forgot to do a fresh pull before running the requirements locally and opening a PR.

## **4 Where Has It Worked?**

SBE is in use at many companies; some are using it well, but many are using it poorly. It is much more common in Europe than in the Americas, because it was invented in the U.K. and the early experts were all in Europe.

### **4.1 Case Study – BNP Paribas**

BNP Paribas was agile, and had been agile for a number of years when they adopted SBE. They had very good metrics showing where their development teams spent their time; their developers were spending 34% of their time fixing defects.

They brought in a consultant who introduced two methodology changes – TDD and SBE. Nine months later their developers were spending 4% of their time fixing defects.

This change didn't happen overnight; it took training and lots of coaching and several months. SBE isn't a new tool, it is a methodology change and a radical paradigm shift, just as Waterfall to Agile was a methodology change and a paradigm shift. The whole team had to learn to think in this new paradigm before they saw the dramatic improvements that the methodology offers.<sup>2</sup>

### **4.2 Case Study – Raytheon Technologies**

Raytheon brought me in to do SBE coaching and training for a data science team; the team consisted of data scientists on one side and developers on the other side, plus a QA team. The data scientists were the

---

<sup>2</sup> <https://www.linkedin.com/pulse/case-study-bdd-improving-throughput-collaboration-simon-powers> and <https://cucumber.io/blog/podcast/bdd-in-banking/>

customer; they told the developers what they needed and the developers worked to deliver it. Misunderstandings and bugs were common.

I began by training everyone in the new methodology and the new way of writing requirements. We began creating our grammar and our interpreter; there was zero automation when I arrived. As we automated more we had to do less manual testing, thus freeing up more time to write automation. Roughly nine months later we had a robust, reusable grammar and a robust interpreter; the data scientists could write requirements and they could be fully automated within a few days.

From that point bugs were essentially non-existent and the developers were able to shorten their release cycle to monthly releases. Over the next year they delivered into Production two defects – and everyone agreed that those two defects were not in the requirements. However, the developers said they should be counted as defects, saying “We should have known that the data scientists would want this”. When the developers *claimed* a defect; I didn’t argue!

We also had one release where the data scientists came back and said that we had delivered a bug. We looked at their bug, went back to the requirement, and said “This is the requirement, and when we run it it passes.” The data scientists looked at the requirement and said “Oh... we wrote the requirement wrong; we will be more careful next time.”

## 5 How Can We Do It?

### 5.1 Understand That SBE Is a New Methodology

SBE is not a new QA tool, nor is it a new Development tool; it is a methodology that requires buy-in from the whole team in order to work well. The team must include whoever writes the requirements in your company – Business Analysts, Product Owners, or Product Managers. Because of this, I teach my clients that they should absolutely ban use of the terms ‘BDD’ and ‘test case’. I once had the manager over the BAs tell me “If you are writing test cases, do whatever you like, but don’t involve us.” I explained that we were writing requirements using the Specification by Example methodology, and that we had to do this in order to deliver automation in-sprint, and the CIO really wanted us to deliver automation in-sprint. At this point the BA Manager had to be involved – he couldn’t allow us to write requirements without his team. Once we had his buy-in, then we could begin a real methodology and paradigm change.

The methodology differences are illustrated in these two RACI charts – in traditional Agile, only the Product Owner is responsible for the requirements. In the SBE Agile methodology the entire team – PO, Developers, and QAs – is responsible for the requirements. This **doesn’t** mean that they must all sit together to write all of the requirements. It **does** mean that they must:

- 1) All agree on the language – the formal grammar – that they will use to write the requirements. If they are creating new grammar for an entirely new requirements domain, then they should probably all sit together to do this.
- 2) All agree on the definitions of terms that will be used in the requirements – for example, they should define what an ‘Active’ customer account means, and how it differs from an ‘Inactive’ or ‘Suspended’ customer account.
- 3) All review and accept the requirements. If the PO writes the requirements, the Developers and the QA team must review and accept them. If the QA team writes a requirement that they believe should have been included, the PO and the Developers must review it and approve it.

Agile				Agile with SBE			
Tasks	Product Owner	Developers	QA	Tasks	Product Owner	Developers	QA
Write Requirements	R,A	I	I	Write Requirements	R,A	R	R
Write Code	I	R,A	I	Write Code	I	R,A	I
Write Test Automation	I	I	R,A	Write Test Automation	I	I	R,A

## 5.2 Bring In a Coach

A methodology change like this can be successful without a coach, but it can deliver value faster with a good coach involved. A good coach can help you avoid the common mistakes and deliver real value faster. I teach all of my classes “The first time we write Gherkin, we all write bad Gherkin. I know, because I did!”. A good coach can shorten the learning curve, quickly getting the team past the initial ‘bad Gherkin’ stage and to the stage where SBE is saving them time. Saving 30% of a developer’s time – perhaps \$50,000 per developer per year – will pay for a coach in just a few months.

## 5.3 Do a Proof of Concept

Don’t try to move the entire company to SBE at once – start with just one or two teams, preferably teams that are starting on an entirely new product. Starting from scratch lets us write the requirements and build the automation in parallel with the code; attempting to convert an existing project to SBE is much harder. If we are starting from scratch everyone knows we have to write requirements, so it is relatively easy to write them using SBE. Then we can start automating long before the developers begin delivering anything to Production – and early automation delivers the greatest value to the developers.

## 5.4 Expand From There

Once the POC is successful, expand to a few more teams. Consider expanding to a team that works closely with the POC team; the POC team will be able to help bring them up the learning curve. Moving a couple of people from the POC team to the next team is also a great way to expand; you are seeding the new team with someone who already has experience. They have learned how to get through the hard parts and can share their knowledge with the new team.

# Team Leadership in the time of remote silence

**Michał Buczko**

Software Quality Manager at Viessmann Climate Solutions

[buczko.michal@gmail.com](mailto:buczko.michal@gmail.com)

## Abstract

In the last few years a huge change in our industry was observed - it was the enablement of Remote Work. It was accelerated during the pandemic and a lot of changes had to be rapidly implemented by IT organizations. Have they been done the right way? There was a huge impact on the way of working in IT project deliveries - office lockdowns later moved to office restrictions and post pandemic restrictions. It was a huge hit to many IT companies, most of the employees had been sent to work from home. Many team leaders had no time to prepare for new circumstances in an organized way. Some of the organizations in parallel with the restrictions found opportunities to evolve, remote work allowed searching for remote talent. Some companies just approved this way of working without any process or organizational changes. Hoping that their employees and team leaders will manage it themselves.

As a result of that, many leaders who were great in management “by walking” or kept pulse with chats in the kitchen over a cup of tea, faced a brand new world of remote relationships and remote conversations. They had to find a way to build teams and lead them in not remote first organizations, but keep connections sparkling with remote work delivery.

Let's have a look in a retrospective how a team leader could support the transition to remote work and changes it triggers. Lets look and analyze practices and lessons learned along the way. Discuss challenges and setbacks encountered while adjusting to remote leadership, including communication breakdowns, productivity dips, and team morale issues. Finally, let's try to search for the future of remote leadership, highlighting opportunities for growth, innovation, and collaboration in a post-pandemic but still remote world.

## Biography

*Michał is a Competence Lead for Software Quality Assurance at Viessmann where he is the advocate for focus on the Quality Processes of Deliveries. He is a fan of fast and efficient testing. Michał gives his insights in design of the test harness and solutions for industrialisation projects, the definition of the scope of testing and efficient test reporting. Michał feels responsible for sharing “new-fancy stuff” with others and helping people grow as individuals. He is involved in various activities including quality management, project consulting and requirements analysis. Now sharing his experience as a Quality Management Consultant. In his free time Michał spends time on archery, feeder fishing or enjoying a glass of rum.*

## 1. Introduction

The remote work enablement, accelerated by the COVID-19 pandemic, marks a significant shift in the way we work, almost as big as the industrial revolution's transformation was in the past. As technology advanced and new tools grew in the 21st century, remote work slowly gained traction, but it wasn't until global lockdowns forced businesses to adapt that it became a norm. This paradigm shift has redefined work-life balance, productivity, and global collaboration, reshaping the future of work permanently.

### 1.1. Did leadership change?

In recent years the landscape of IT delivery related work has undergone a huge shift, with remote work impacting the modern workplace. The transition to remote work has forced organizations to rethink their way of interaction with employees. The companies had to adapt their policies, infrastructures, and cultures to accept distributed teams. Teams also have been reshaped by the remote work paradigm. There was a dynamic change of virtual teams to enable fostering innovation and ensuring productivity in the new situation. Teams had to rebuild the strategies to build trust, camaraderie, and a shared sense of purpose in a digital environment. The most significant transformation has been observed in leadership. The evolving role of leaders in a remote work context required leaders to have empathy, flexibility, and digital proficiency. Effective leaders had to navigate the complexities of managing distributed teams, ensuring employee well-being, and driving organizational success in a virtual world.

### 1.2. Context of the observed team and organization

Here we will present some of the challenges that have impacted software delivery work in recent years. It was possible for us to personally observe the change at Viessmann Climate Solutions organization, who works on building HVAC devices and IoT platforms for its customers. It had moved through a revolution in the last few years moving from 9 to 5 office focused work to a multi site, remote-enabled organization. This required not only the work shared across the offices but management silos in the HQ to be broken. Many roles were globally impacted from remote sites, not only decisions made inside the HQ. In addition each of the sites working across Europe had to implement enablement of remote work and remote teams of their own, requiring their leaders to build a new model of cooperation and managing distributed teams.

## 2. New way of working

The remote work revolution has not only changed where and how software delivery work is done, but has also reshaped the organizations, teams, and leadership models. Going through this new landscape, it is clear that adaptability, innovation, and a human-centric approach will be the keys to success in the remote digital age. Organizations, teams, and leaders have to utilize the power of remote work to drive progress and achieve new heights.

## 2.1. Switching on

One of the first challenges that is being faced by the leaders and team members is the “switching on” ritual. In the “old” office days, work time started in the office kitchen with the other employees in a small chat. Moving more towards project topics and with the cup of coffee we got our desks and felt ready for the work to start. In the new remote days everybody has their own routine. Some start the day with cameras off in pajamas, some with breakfast on their desk. It is not common to dress-up, eat breakfast and travel to the office anymore.

At the Viessmann Quality Team there was a practice where the team starts a day with a free to join meeting, where any team member can touch any topic they want. It is allowed to have a cup of coffee or eat breakfast while talking with others. This is the way to begin the morning together and switch on for a work day. The meeting is open for discussion about anything from world news, company questions, today's tasks, private life, today's mood or project gossip. The goal of the meeting is to get ready and leave the call once we want to start daily operations. It is important to not interfere with project teams' daily stand ups with such meetings, or have a clear priority here. The meeting is set as optional for all the invited members, due to being the lowest priority against other project activities. It is a good approach to have it before standups, if possible. From the leader's perspective this meeting is a good place to refresh with the team their plans and challenges for the day from both project and private perspective. It can be used to update priorities or answer any puzzles and questions the team has, regarding a recent event or just after over the night thinking on some issue. It can be used for a fast feedback loop, pulse check on team spirit or just be an insight to the mood barometer as it was with the start of the day coffee and chat in a kitchen. It can be also used to adjust the plan of the day with sudden unexpected meetings or changes.

## 2.2. Switching off

The “switch off” for remote work is almost as tricky as the start of work. In the office there was a point in time where people just stood up and left the office. Now the home is the office; it is easy to lose track of time and get focused and want just to finish this one more small task and sit by the desk till the late evening. It is the role of team leader to help the team to keep their balance. Some people take breaks during the day as they know they will work late, but others do not have to join them with hidden overtime hours.

One way of pushing the team to achieve their balance is mention their off work activities during afternoon calls and make sure they have the time to practice them. Leaders can complement team members' successes and achievements. Push the team to openly talk about their after work activities:

- Try to embrace their exercises and hobbies
- Support spending time with family and friends
- Talk about their goals and pursuits
- Ask them them to speak about their achievements loud

It is important for the leader to show the team how important it is to have success not only in professional work but also in their private activities and try to join with the whole team to celebrate them. This makes it a lot easier for everyone to embrace their hobbies or sport activities and to remember about balance and other plans. The object is to create an atmosphere where everyone feels that it is okay to switch off and finish the day even if others are still working.

## 2.3. Building work related and off-work relationships

One of the most important aspects of daily work for the new remote leaders is to build trust and relationships with the teams. They are no longer inside the office and company inclusion and relation to the place of work is a lot more shallow. The only way to compensate for that is to have a team spirit inside the team and relationship with the leader or manager. It is the last space that can affect the team members' decisions and feelings about the company. To earn this trust, leaders do not only have to be open and proactive toward the team. The open communication has to be focused on building the relationship inside the working time and outside of the working time. Sometimes integration events or team building might happen in the evenings or on the weekends, often in remote locations. If the leader loses it, there might be no chance to build it back. Also allowing the team to schedule a meeting with the leader and the leader's availability to discuss any topics that are required. Leaders have to be ready to speak not only about projects and performance, but also feelings and fears that their teammates have inside the work and maybe also outside of work. Leaders have to be understanding and open to talk about private lives. Remote work blurs the lines between daily life and daily work and the more a leader becomes a part of the life of their team the better they know them and earn more of their trust.

## 3. New conflicts in remote world

This new age of remote work brings to the table a whole new scope of challenges and conflicts that both employees, teams and leaders need to tackle. Many of those are still being investigated and best practices are needed. We need to find the best way to resolve them and make our work space friendly again.

### 3.1. Balancing work and family

The first and possibly most important conflict is work and family imbalance. In the past we worked in the office and our family was separated from our work space and daily business. Now people work at home, they want to keep the professional approach to work, but at the same time many of their homes were planned for family time. For example in Europe small flats are common and there is no possibility to separate a work space for most of us. Team members often do not have a separate space or additional empty room where they can fully focus on work. Often they try to utilize shared rooms, and cope with home responsibilities and expectations from the family that they can fit in in the daily schedule. Many of our employees need to share the professional work we do for the rest of the day with family members who might be in the house at that time. The split between things that happen in the house impacts the work time. It requires an art of detachment at a high level to allow yourself to focus on work while kids play behind the door or

across the room. Some of us need to manage the time for family duties and work, so elastic work time can help us. We do not have to fully focus on 9 to 5 work. We can keep core hours available and choose the best time and possibilities for the rest of the day.

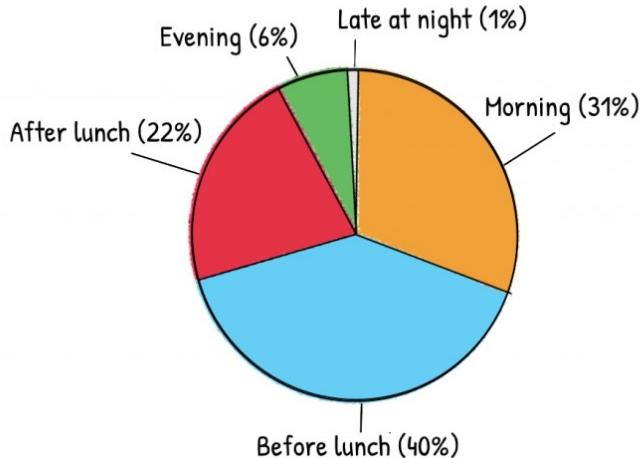
As a leader we need to be understanding and supportive for the team. It is not the lack of professionalism or bad will of team members that has the background noises of living home in the background. One of the practices that can help organize time and team workflow is to encourage team members to openly push their private appointments and duties to the calendar. It builds a clear image of the daily plan. It enables the possibility of not having to dial to a meeting from a car or shop, just because it overlaps personal responsibility. It creates a culture to share with other teams openly responsibilities and timeframe and move meetings that are planned in not perfect time.

Calendars from Gmail and Microsoft recently have the option to set office hours availability to give other coworkers insight to our work schedule, availability for meetings and ad hoc discussions. Remote work supports and thrives on giving team members the responsibility and freedom to define their own way on work planning and executions. Leaders have to be demanding on deadlines and goals being kept and achieved, but should not push for a strict working schedule or task micromanagement. Many of the employees have kids at home during summer interfering with work during the day and have more free time in evenings and late afternoons. The leader should encourage the team to build their best work schedule that would ensure work is being delivered and keep the availability visible for other teams in a transparent way.

### 3.2. Workdays and weekends

There were many studies made on people peak performance time, and as expected “it depends”. Each of the employees is different, have different home work setups and family at home situations. Remote work allows the teams to adapt the time frame of the day for their own best. The leader pushes the team to evaluate each of them individually and share results inside the team. With that information, the team can choose the best way of work for them.

During what time of the day do you reach  
your productivity peak?



Source: TalentLMS Remote Work Survey

As it was mentioned before, it is not the old days with 9 to 5 workday anymore, we have a whole new remote rule. We need to keep the core hours available and plan the day with the impact of work and family. The most crucial part of it is transparency and visibility. Give the team responsibility on the targets, communicate clear time frames and give freedom on execution. In some teams the weekdays and weekends became more fluid as the relics of the old office times. If anyone on the team has something urgent to take care of during one of the weekdays, we enable them to plan some time during Saturday or Sunday in exchange. This way until they keep the 40 hour work week they have the freedom of adjusting their work time to time slots when they can work without being interrupted. The main focus is to do it in a transparent way and make it invisible in their calendar.

The team leaders have to focus on the defined expected results, deadlines, and timeframe to achieve it, i.e. 40 hours working week in Poland. It requires them to give the team members freedom to use their time the best way they can. Remote work is based on trust, as long as the team is transparent and open about their absence and their work availability time. This way teams use the freedom to build engagement and better attitude and professionalism. Still if someone abuses this trust or does not behave professionally with the freedom, leaders have to have measures in place and be strict and consequent with repercussions.

### 3.3. From Face-to-Face to One-to-One meetings

In the old days the baseline of communication between the leader and the team members was a set of face to face meetings with each of the team members. The frequency was dependent on the size of the team. The current remote work times stopped the office meetings and the “leadership by walking” practices. One of the first changes was the full switch to remote and online

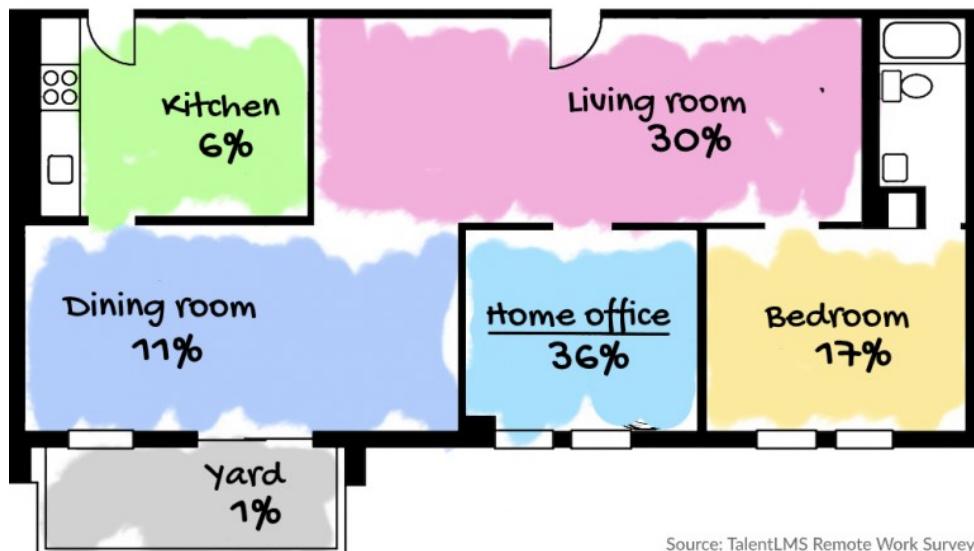
meetings. It also started the change in the naming convention and those syncs with peers are often called “One to ones” as they happen in the virtual world now. The goals of them also got extended, as they are not kitchen space or quiet room meetings for small talks. The one to ones are a lot wider, they are the main point of the relations and communication between leader and the team. A good practice, if possible, is to run them on a weekly basis and to allow your team member to be the owner of the topics discussed. As leaders, we have to be open to discuss their private and project related topics or just today's mood and personal feelings. If there is a need for feedback or performance review related discussion it should be moved outside of the weekly sync and delivered at a separate, dedicated meeting. It is a good practice to propose or be open to some alternative options of executing such a meeting. Both people do not have to spend the call sitting in front of the cameras, they can be run with alternatives like joined remote walk with being dialed on phones; another idea is to try to run a quarterly sync on lunch together in some remote location halfway between yourselves or meet each other over a dinner on a trip together. Please remember that this meeting is meant for building and maintaining a leader's relationship with the team, but also is the crucial connection between the employee and the company that leader represents, as most office related connections are not there anymore.

### 3.4. Home versus office - working space challenge

Another challenge for the employees that team leaders need to be supportive with is the work spaces for our teams. The remote switch was unexpected by many organizations and employees and the homes and especially the flats often do not have a dedicated room or even space for work. In addition the company's approach to remote work start was done in various ways. Some of them gave their employees some IT equipment or additional money to cover higher bills, less of them offered vouchers to order a desk or office chair for a home office, but most just waited and did nothing to support the employees. It is the leader's responsibility and challenge on a daily basis to be respectful and understanding if someone doesn't have a separate space for work at their home. Situations with family members and home pets will happen during meetings. Please remember that employees' home spaces were designed to fulfill their private life needs and remote work was often not considered. That is why work is just an additional not planned scope that needs to be handled there and not many of us have houses with backyards that could be extended by an additional room or studio for work.

Leaders have to remember that not all of us have the same home space capabilities and some turned off cameras are just triggered by surroundings. The leaders should not force the camera to be active, but build a space in which people feel safe on a call and will be willing to have them running even if working from the kitchen or children's rooms. The surveys on the remote work spaces give insights on the situation as no room is safe from not being used for work:

## What part of their home do remote workers work from?



Source: TalentLMS Remote Work Survey

The team members have to share the flat or house with different numbers of people or have different family sizes. Sometimes some of them also work remotely. Sometimes children have off school holidays. Home is primary for the families and they use it every day. The remote leader role is to be respectful and supportive even if someone interrupts during the meeting. It can be used as a possibility to extend relationships and to know better the teammate's daily life and family. Please do not build unachievable expectations of strict professional work space and isolated working time in the middle of someone's house. This was never the goal for which this home was designed. If the leader recognizes that one of their peers struggle, they could try to use their voice inside the company and search for an opportunity to help that person, maybe with a voucher for co-working space or additional equipment for their home-office setup that could help.

## 4. Meeting overload

Meeting overload is another huge topic affecting the remote teams. In the office meetings were always limited to the work hours, availability of meeting rooms etc. In the remote and virtual world there are no limits. It is the leader's responsibility to ask the team to be very assertive in accepting invites that are needed for the business or work. Let them be restrictive in the topics selection, or to reject empty invitations looking like unprepared meetings. If those rules are not followed, there is a huge risk of ending up with an overbooked calendar full of overlapping meetings with no time to do the actual work. This problem is clearly illustrated in the example below:

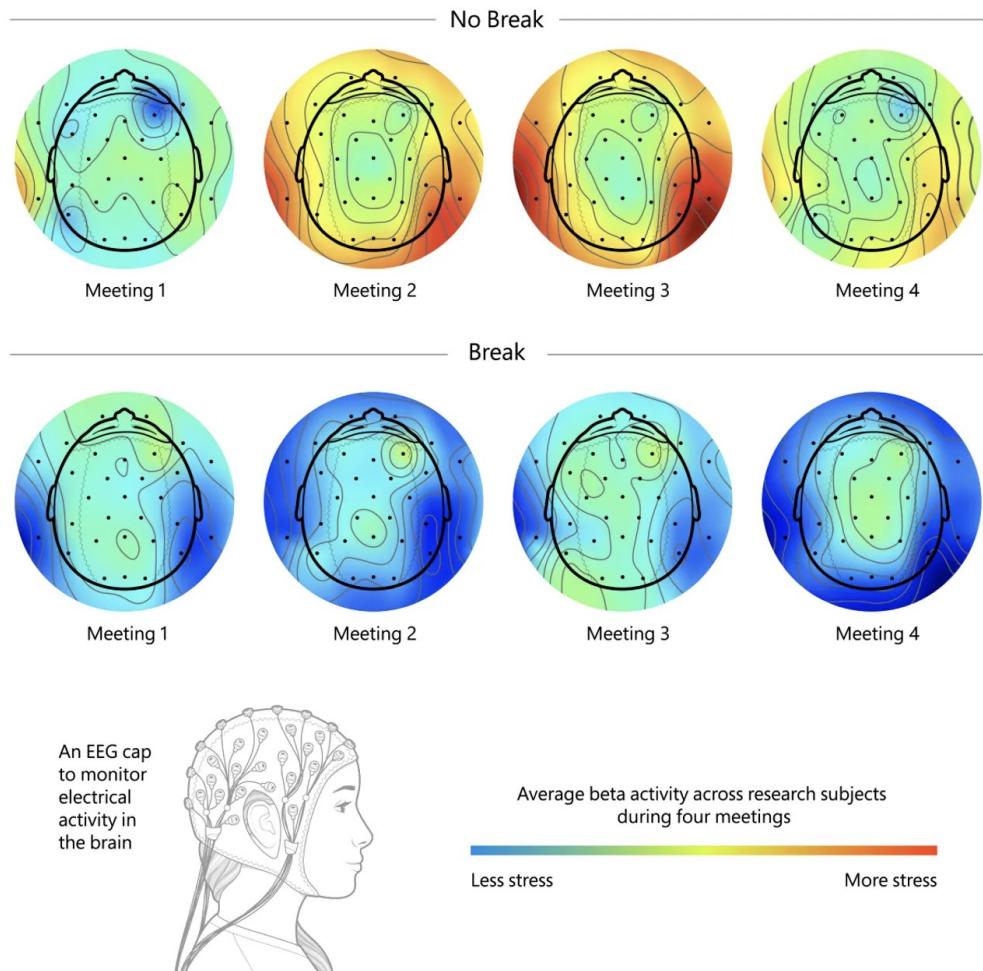
Handoff: Showing app 09:00 – 10:00	Review roadmap notes 09:00 – 10:00	NO MEETING WEDNESDAY so they say 09:00 – 11:00	Customer 09:30 – 10:30	Working session, 09: Review model 09:30 – 10:30	Interview debrief, 09: Edit customer interview 09:30 – 10:30
Draft OKRs 10:00 – 12:30	External API, 10:00			Leads planning, 10:30	1:1, 10:30
	Feedback on figma, 11:00			Launch demo: Deliver OKR estimates, 11:30	All hands 11:00 – 12:00
	Apparel brand planning 12:00			1:1, 12:00	Customer interview, 12:00
	Apparel retailer planning 12:00 – 13:00	Apparel Team Review catalog doc, 13:00	Interview 13:30 – 14:30	1:1, 12:30	Interview, 12:30
Experiment Review, 13:00	Feedback on survey 13:00 – 14:00	Product Team review 14:00		1:1, 13:00	1:1 13:00 – 14:00
Ramp: Bulk Uploader 14:00	Finish doc, 14:30	Weekly Ramp Edit OKRs 14:30 – 16:00		Review survey results Interview, 14:00	Review survey results Interview, 14:30
1:1, 15:00	Chat Apparel roadmap 15:30 – 16:30	Interview, 16:00		Review workstreams 14:30 – 17:00	Draft leadership update 1:1, 15:30
External API, 16:30	Edit OKRs, 16:30	Apparel weekly sync, 17:00			Plan next week 16:00 – 17:00

Try to use plugins for calendars and set up some ground rules for the calendars. Remember about breaks between meetings, coffee and eating slots, or have other duties come up, they need to be added to the daily plan. Employees can also show their time availability in the calendar to make it clear when he or she can join and help the team resolve the subject.

If someone requires a deep focus time to perform technical work, please put it inside the calendar. Make sure that transparency and visibility are the basics of the calendar management and share it with coworkers. Be restrictive on those focus breaks and not make exceptions on accepting and overlap meetings on those sessions. This should be the time to have a single thread focus and fully work on delivering it. Many of the currently existing calendar applications and time management tools have a lot more options than just the meeting scheduler. The leader and the team should invest the time to know the tool better and investigate features and use them, as they are a great help to organize remote work.

Sharing a calendar is a nice practice allowing others not only to see the free/busy slots but also the details of booked time slot goals and decide if the cause of the meeting has bigger or lower priority over that. There is also expectation on the part of the leader to be respectful of those “deep focus” sessions or be clear on priorities in planning overlapping meetings and asking the team to join. Transparency builds trust and leaders should not use it in a wrong way. It is also important to remember that “deep focus” sessions do not have to be spent by the computer or at the desk in the home office. Many people just take a walk to think about a subject and make a decision. They can record or make notes on a mobile device and later push it to the computer. The leader has to be respectful and give freedom and responsibility to the team of how they use those sessions and not always expect them to be in front of the monitor.

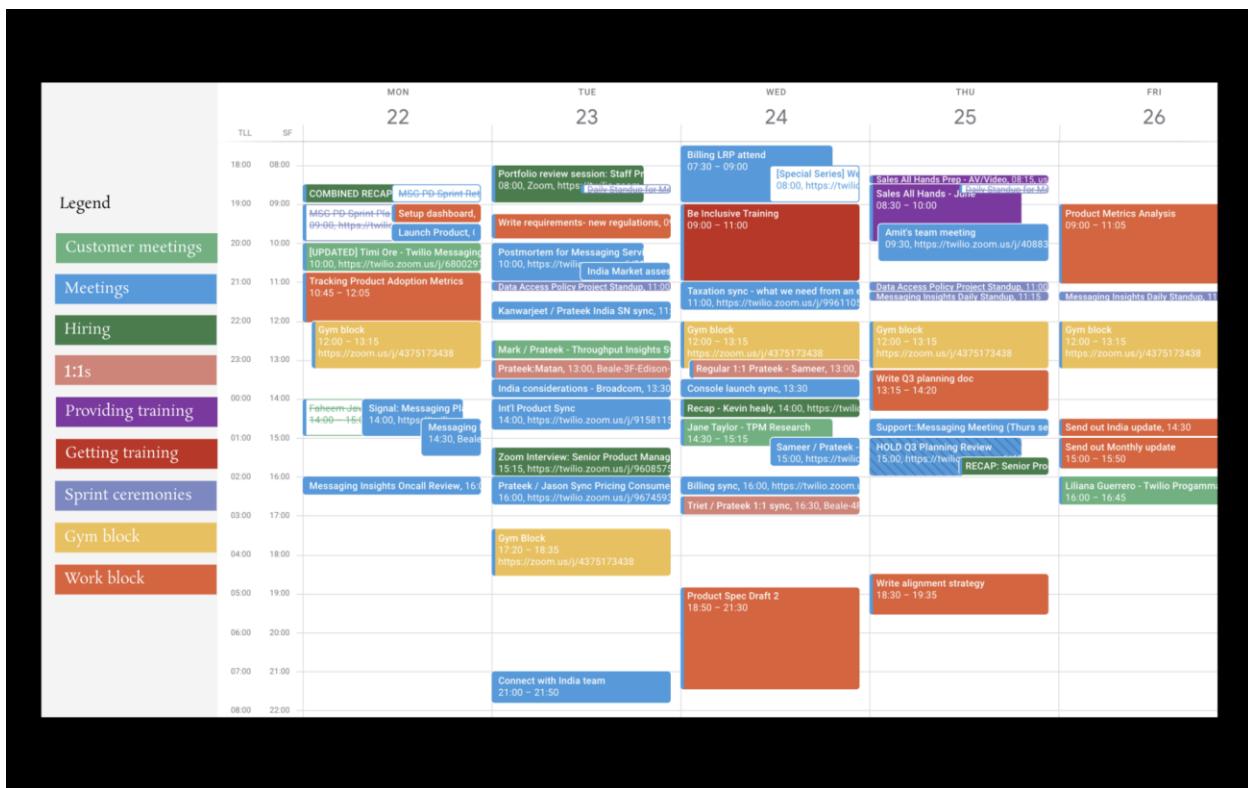
Another subject growing in recent remote times is “decompression time”. People in our workplaces love to push a meeting session right after another in the calendar. There were multiple studies to discover how a continuous meeting marathon affects the brain. Thanks to the monitoring of electrical activity of the brain it was discovered that people who managed to have a decompression slot between the meetings had a lot more focused and less stressed mind during those meetings. The results visible on graph below the results are stunning:



No breaks end up with huge stress and a fast path to work burnout. It takes much more time to switch off after such a day and it affects and even breaks the balance of our work/family life. That is why the leaders need to work with the team members to help them manage their time, avoid meeting overload and if the calendar extensions or additional tools support the “decompression time” capabilities in the calendar, ask the team to use it at least as 5-10 minutes slots for the hygiene of the body and the brain.

## 5. New way of remote teams working

All of the mentioned challenges and aspects forced the remote teams and their leaders to search for a new working model and practices that could enable better communication and efficiency of the work done. One of the first big changes is that the calendar became the main focus point for every plan we do: meetings, focus sessions, breaks, work time availability, private responsibilities, all of it should be there. Remote work requires strong trust and relationships inside and outside of the team. The best way to achieve that is transparency and openness in our daily routine. The work calendar is the best place to build it.



It gives the user many features and automated rules configuration options to make it more open and transparent, but also efficient and healthy. Later the calendar can be shared and its content visibility be managed. With the embrace of the decompress time and the switching on and switching off sessions it can help to build healthy work space space for remote work with full visibility. Here is an example of a transparent calendar day below, but it still lacks decompression sessions and would be seen as overloaded:



It is not just only the calendar management role that gets extended, there is also visible a large communication shift. In the old office times the rule was, if it's urgent just go to someone's desk, if the person is not there just write an email. That was the rule of communication, online communicators were used mostly for off topic discussions or urgent requests to someone sitting in a meeting room.

Nowadays it looks like nobody reads emails, they are used only for formal communication we want to have recorded or just as a method to push invites to someone's calendar. If someone wants to ask about something, but it can wait a while, they use communicators like Teams, Slack, Google Chat, but nowadays it is not expected to get immediate response there. Most urgent cases went back to fast mobile call, or online huddle or any other direct communication with voice or video. Most of A-sync tools are forgotten, the ones that were the immediate ones now are treated like a-sync. Nobody has the time to check emails, they wait for weeks and months for response or being read once someone is asked about them via communicators or during a meeting. In parallel lots of web based alternatives for Outlook Express that were a standard in the past don't have the same capabilities and mailboxes are not readjusted due to pain of the tools itself. Employees can review and respond to the invitation in the calendar without looking at the mailbox. A huge risk is being found during audits that most verbal agreements do not have a record in a form of document or action item, even if delivered, as no track record of it exists. Not all companies can operate in such a verbal and trust based way with, for example, ISO requirements fulfillment. The leader has to be aware and ask someone in the meeting to make at least a formal note of it and summarize all the decisions.

## 6. Team leadership in remote work

The remote work shift not only affects the teams but also reframes the role and responsibilities of the leader. One of the main points is the “glue work” to keep the team together, focused on shared goals and a good atmosphere. In the past many of those integration and off work activities were managed and planned by the HR department with line management directly. Now many of those are shifted to the team formal and technical leadership. It is their task to plan, organize, budget and deliver those collaboration activities for the team inside or outside office hours. Company wide parties are symbols of the past. Now the integration needs to be team focused and the leaders have to be creative, with budget limits, distant locations of team members to organize in the middle of the distance activity, or some new online interactions that would help the team to build a positive team culture and atmosphere.

The new wave of remote leaders have to be also good advocates of their teams, showing their work and making it visible inside the organization. Both visibility and recognition in remote times is not so easy and leaders can use their direct access to management to build relationships and present the successes and work done. They can also make tighter feedback loops and collect both recognitions and suggestions on the work done to keep not only bottom up motivated improvements, but also help the team identify aspects that can make the work better and more visible organization-wide. It is the leader's role to enable the visibility and continuous relationship between team and up the chain, making sure not only top individual achievements are being recognized, but that team effort and its fruits are also appreciated.

Another shift is the responsibilities of the leader role expected from both company and team sides. The leaders are seen by the organization as their voice at the team level, it is more and more dependent on them how the message from the company will be shown, received and understood by the team. It is them also who needs to collect the feelings and thoughts of the team after the change was announced.

## 7. Summary - Remote Leadership redesigned

Remote work has shifted the team leadership model at both organization and team expectation levels. In the Company contexts, new remote age leaders have to be the voice of management to support the corporate message and clarify the scope required. It is also now the leader's responsibility to collect the feelings from the team and decide what to push back to management as team feedback. The leaders should also build and keep the trust and open communication inside the team not only about team internal subjects but also topics outside of delivery scope focus like business situation, company news, and strategic shifts or departments reorganization.

In parallel also the team's expectations of remote teams are a bit different as the leader is usually the only person between remote employees and the organization itself. Therefore the leader is a representation of the team. To make it a better atmosphere for remote work, support leaders have to keep the trust in the team, maintain open communication, and give the autonomy to the people.

They need to be more results oriented than company duties and work time focused. Teams need the responsibility and trust that they commit to the team goal and the a-sync work has to be embraced as part of remote scope. The leaders are solely responsible for keeping the trust with open and honest communication. They need to address all the appearing topics directly, bringing people together on offsite remote meetings or online appointments. As the company communication is not so direct anymore, all the building, supporting, and affecting of peoples' careers and expectations have to be driven by the leaders who know the employees best.

It is a shared responsibility of the leader and the team to build and support the team's shared identity and team spirit. Leaders must support their people by allowing them to express both professional and private matters with them. If an employee needs that, we should not try to redirect people to HR or a People organization too fast as now we know our remote team friends best and should be the first line of trust and discussion. The role of leadership has grown to be a company wide first level representative for all the employees from the team. For team leaders to grow in such a role, they have to embrace these remote team leadership and team-building tenets as critical components for success of their teams, their companies and themselves.

# SQA skillset, AI and Advancing medical research

**Tracy T. Burridge**

tburridge@gmail.com

## Abstract

Software professionals have a social responsibility to utilize our skillset to help advance medical/scientific research. SQA discipline and skillset is uniquely positioned, as we do not build tools, we utilize tools to push boundaries, to verify declared limits; to discover software failings that coders and designers did not imagine, love the phrase “why would a user do that?” We, SQA, reside outside of the box, we do not just want to break things, and we want to describe what we did, document our steps and show that it’s repeatable.

Arguably, Y Combinator is one of the best known and most respected incubators. In February 2024, YC published a list of “Start-ups we want to fund” innovative entrepreneur requests. On the list is “Cure cancer, the technology we’re talking about is an MRI. Modern MRIs are sensitive enough to detect cancer masses as small as a millimeter.” Kaggle has been offering bounties for a few years now. Analyzing data, approaching all publicly provided data with all skepticism - exception boundary testing, edge cases - SQA/SQE professionals know how to do this better than most other professions.

ChatGPT, Knowledge Graphs and other AI tools, can be used with publicly available medical datasets to advance medical challenges such as cancer or autoimmune diseases such as multiple sclerosis -MS. What is fascinating about MS is that we, in 2024, still have no idea what causes MS. There are theories, there are pharmaceutical efforts that pause or delay the progression but are very very expensive.

***What if I use my SQA skillset, AI tools, and available datasets to establish a set of repeatable steps to establish new methodology for early detection of MS?***

## Biography

*Tracy Burridge has 20 plus years in software quality engineering from start-ups and enterprise software companies. She has earned an MBA and MAT, experience in developing software requirements, SQA frameworks and training/executing tests in environments from Gaming, databases and UI heavy applications. Currently, at TriMet as a Fare Systems Analyst and has worked on an assortment of projects including smartcard transportation upgrade, research and requirements gathering to update internal POS systems, Federal Transportation IMI Grant and the updating of all TriMet customer facing systems - app, website, Inspection and ticket machine software.*

## 1 Introduction:

A good software quality engineering has an understanding of a successful test cycle. Not just finding a defect, but the ability to drill down and communicate the finding. When a bug report is submitted, it will contain steps of repeatability, common identifiable factors, and patterns.

A person, a human being, is for all intent and purpose an extremely sophisticated, complicated program. Like any program, there are defects, issues that are sometimes explainable and other times unexplainable. In either instance if there is a defect, we can or should be able to identify overlapping environmental factors, steps to reproduce the defect, rate of repeatability and predictable or expected outcomes.

We evaluate a defect for causation vs. correlation. As SQA we sometimes, often times, do this in a vacuum; we spend a lot of time explaining how we found the defect and how the average person on the street “just needs to follow the step-by-step instructions and they would be able to generate the same defect.” Sometimes this is a showstopper, often times there are work arounds.

Today, however, we have new tools at our fingertips – a near personal assistant or eager college intern, technologies such as ChatGPT 4.0, WikiData and Google AI. We can introduce the defect to these tools, use traditional SQA/E practices to assist in advancing research in ALS and MS prevention/detection/repair.

## 2 Approach:

At the start of this endeavor, I was quite uncomfortable making the necessary shift of Person to Software Program and Disease to Defect. If I want to use my SQA skills to address the very human disease, the adjustment was necessary, so onward ho!

- Evaluate and select disease:
  - Cancer: I chose not to evaluate Cancer as it is broad and has an exceptional number of research efforts.
  - **Multiple Sclerosis MS: MS is predominantly a female disease, 1:300 persons are afflicted (1)**
  - Lou Gehrig's disease ALS: ALS is predominantly a male disease; 1:400 persons are afflicted. (2) It is a sclerosis like MS, but it has the advantage of an amazing speech made by a beloved athlete:
    - Lou Gehrig “For the past two weeks you have been reading about the bad break I got. Yet today I consider myself the luckiest man on the face of this earth.” (2)
- Tools: Mindmeister, ChatGPT 4.0 and Google AI - to narrow down large datasets or organize dis-organized information

- Define the defect: Description, environment, expected and actual behavior. Rate of repeatability.
  - Incorporate the call out, data organizations, such as Kaggle and the incubator Y Combinator, have started making the call to software professionals to start engaging with science research to use publicly available data/datasets to help find cures to diseases like Cancer, MS, ALS, etc. Countries like the U.K. have already integrated the two professions. These vetted and solid resources for reviewing analysis are underway, allowing for comparisons and peer reviews.
  - Observations and recommendations
- In this situation, the objective of a mind map is a way to assemble documentation to create Requirements – remember, we are QA and if we do not have requirements, we make requirements.

### 3 Define the disease (defect) that will be used as a Use case

A customer has reported “Bug”

- a. What is the: Defect” - Multiple Sclerosis / ALS (Lou Gehrig’s disease)
  - i. MS affects women
  - ii. ALS affects men
  - iii. Both are a Sclerosis - with unknown causes
- b. What is the process to isolate and describe the bug? Develop and fill out a Bug report (draw from “best practices”) See appendix
- c. Select and define the QA methodology that will be used to approach reported bug

I will take on the role of “disease tester”, I have been given a program, the human body; I have been informed that the program is very complicated and for the most part very reliable and very predictable. While the human body is a solid program it will, on occasion, turn on itself – the ordinarily reliable program, will, without predictors, progressively stop working. The defect has been given the name sclerosis.

In QA, if there is documentation, ideally a detailed requirement, the first thing I would do is read through and develop a plan defining test cases, scenarios, user flows and user stories. In this case, I know what the “ideal” human looks like – disease free, immune system not gone array, a software program in which everything works perfectly.

**Because of the complexity of diseases listed above, I chose to focus on MS for the following reasons:**

- There are many theories that Vitamin D deficiency is a potential contributing factor
- Vitamin D deficiency is an established condition
- Vitamin D can be researched
- MS is a disease, in comparison to ALS, with theories

As a Tester, I am given a program and told that it will:

- Not run optimally in 1 out of 300 installs

- Once the poor performance starts the program can never be reset
- There are tools that help the program to run as close to optimal as possible – still not optimal

My approach is not going to be that of an SQA Lead (planning/proactive), it will be as a tester engineer (reactive/ all types of testing). The project is underway:

Tester characteristics defined as (3):

- Reactive to a program that has been assigned
- Focus on finding bugs
- Conduct a variety of test techniques, both manual and automated
- Support the development process

**Table to create understanding of ideal human vs defect in human:**

Ideal/Expected	Defect
Absorbs Vitamin D	Does not absorb Vitamin D from sun or diet
Immune system does not attack person	Immune system over protects, attacks the person.
Muscles grow and regenerate	Muscles degrade and not repaired
Vision is assisted with glasses, not pharma	Vision degrades and cannot be repaired
Walks with ease until a senior citizen or injury	Ability to walk becomes a struggle and only gets worse

## 4 No documentation, how do I trigger the defect?

If I am not provided documentation, I will need to use **ad-hoc or exploratory testing** to first understand the defect. The tricky part with this Human program turns out to be creating these User scenarios as the documentation is sparse, and what is available is very theoretical and there seems to be a lot of inconsistency.

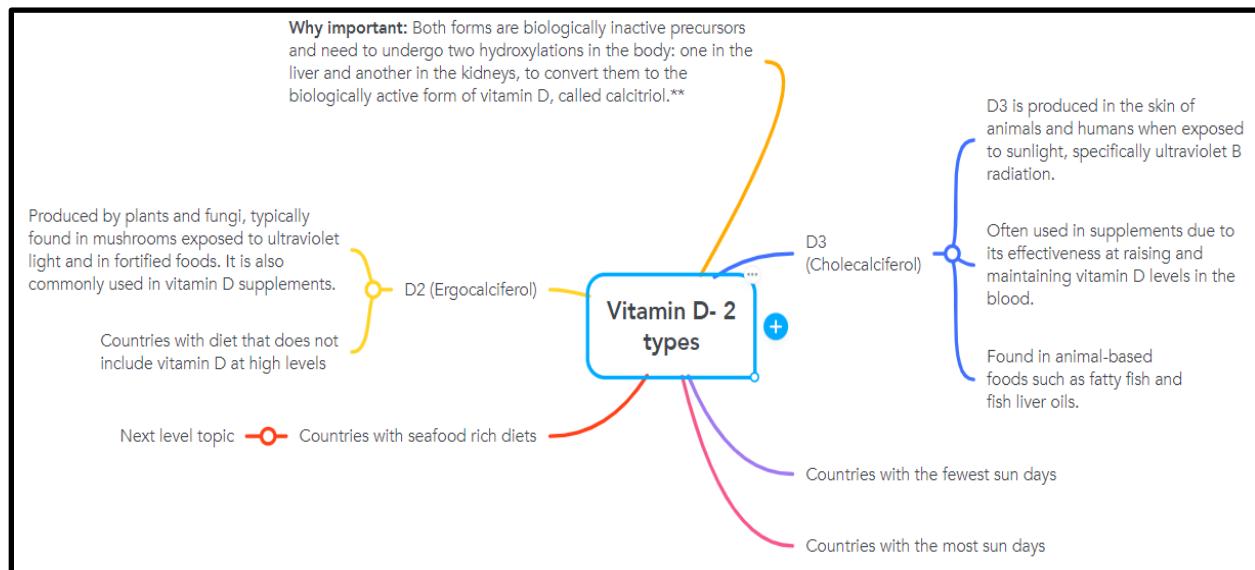
- Uniquely, this defect we are taking on comes with Use ChatGPT to help get through large datasets that are focused on:
  - a. MS
    - i. Autoimmune disease
    - ii. MRI's
  - b. In addition, potentially:
    - i. Recent Studies
      1. International
      2. Domestic
- Use Knowledge Graphs to parse the datasets:
  - a. Pull/Graph the known:
    - i. Validation of current theory
    - ii. Challenge current theory
    - iii. Find the gaps in current theory
  - b. Pull/Graph new findings and theories
  - c. Develop new theories

## 5 Create Mind maps: Disease – Multiple Sclerosis



Figure 1: Multiple Sclerosis at a high level , <https://mm.tt/app/map/3333723184?t=k8HaPlbe4x>

## 5.1 Vitamin D mind map:



Develop a mind map in order to have a quick reference. Here, Vitamin D is outlined as having two distinct components, D2 and D3.

D2: Absorbed through diet and D3: Absorbed through Sun exposure, Both forms of Vit D can also be found in supplements.

## 5.2 Countries with high levels of Vitamin D3 deficiency, ChatGPT 4 vs. Google AI

Source: ChatGPT 4.0

<p>Vitamin D deficiency is a global health issue, but certain regions and countries are more affected due to factors like geographic location, lifestyle, skin pigmentation, and cultural practices. Here are some of the countries and regions with the highest levels of Vitamin D deficiency:</p> <ol style="list-style-type: none"> <li><b>Middle East and North Africa (MENA)</b> <ul style="list-style-type: none"> <li>Saudi Arabia, UAE, Egypt, Iran: Despite abundant sunlight, cultural practices like clothing that covers most of the body and limited outdoor activities contribute to high deficiency rates. Some studies show deficiency rates as high as 60-90% in these regions.</li> </ul> </li> <li><b>South Asia</b> <ul style="list-style-type: none"> <li>India, Pakistan, Bangladesh: High population density, pollution, and darker skin pigmentation reduce the synthesis of Vitamin D. Deficiency rates in urban areas can exceed 70%.</li> </ul> </li> <li><b>Northern Europe</b> <ul style="list-style-type: none"> <li>Norway, Sweden, Finland, UK: Limited sunlight during the long winters and higher latitudes contribute to lower Vitamin D synthesis. Despite fortification efforts, deficiency is still a concern, especially during the winter months.</li> </ul> </li> <li><b>East Asia</b> <ul style="list-style-type: none"> <li>China, Japan, South Korea: Urbanization, indoor lifestyles, and air pollution contribute to Vitamin D deficiency. Studies in these countries report deficiency rates ranging from 40-60%.</li> </ul> </li> <li><b>Africa</b> <ul style="list-style-type: none"> <li>Morocco, Tunisia, Algeria: Similar to the MENA region, despite high sunlight, cultural clothing and indoor lifestyles result in high deficiency rates, particularly among women.</li> </ul> </li> </ol>	<p><b>6. North America</b></p> <ul style="list-style-type: none"> <li>Canada, Northern United States: In Canada and the northern parts of the U.S., the long winters and limited sunlight exposure lead to lower Vitamin D levels, particularly in the winter months.</li> </ul> <p><b>7. Australia and New Zealand</b></p> <ul style="list-style-type: none"> <li>Despite being sunny regions, cultural practices like avoiding the sun to prevent skin cancer contribute to Vitamin D deficiency, especially in Australia.</li> </ul> <p><b>8. South America</b></p> <ul style="list-style-type: none"> <li>Argentina, Chile: Higher latitudes and colder climates in southern parts of these countries contribute to higher deficiency rates.</li> </ul> <p><b>Key Factors Contributing to Vitamin D Deficiency:</b></p> <ul style="list-style-type: none"> <li>Latitude: Countries farther from the equator have less sunlight, especially in winter.</li> <li>Cultural Practices: Clothing that covers most of the skin, particularly in Muslim-majority countries.</li> <li>Urbanization: Increased time spent indoors and air pollution blocking UVB rays.</li> <li>Skin Pigmentation: Darker skin synthesizes less Vitamin D from sunlight compared to lighter skin.</li> <li>Diet: Inadequate intake of Vitamin D-rich foods or lack of fortified foods.</li> </ul> <p>Addressing Vitamin D deficiency often requires a combination of public health measures, including supplementation, fortification of foods, and education about safe sun exposure.</p>
---	---

Source: Google AI search results

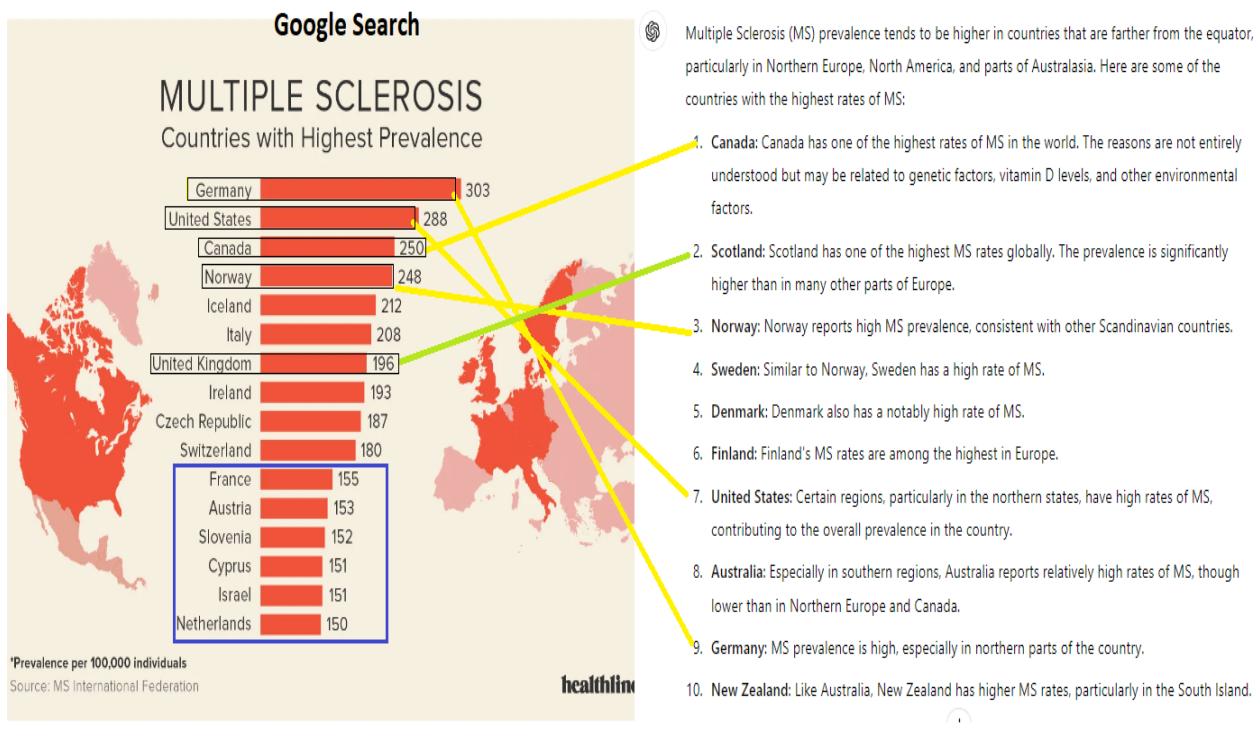
 Search Labs | AI Overview

Some countries with high levels of vitamin D deficiency include:

- Pakistan: Has the highest prevalence of vitamin D deficiency in adults, at 73%
- Bangladesh: Has a prevalence of 67%
- India: Has a prevalence of 67%
- Nepal: Has a prevalence of 57%
- Sri Lanka: Has a prevalence of 48%
- Afghanistan: Has a high prevalence of vitamin D deficiency, with 45–65% of the population having levels below 20 nmol/L
- Iran: Has a high prevalence of severe vitamin D deficiency in infants, at 86%
- Turkey: Has a high prevalence of severe vitamin D deficiency in infants, at 51%

Some reasons for vitamin D deficiency include diet, sun exposure, and darker skin pigmentation.

### 5.3 Geography: Where is MS found on the world?



### Alignment of Regions/Countries with Vitamin D deficiency and High levels of MS

Country/Region with Vitamin D deficiency	Country with High Levels of MS
Saudi Arabia, UAE, Egypt, Iran	
<b>S.Asia:</b> India, Pakistan, Bangladesh	
<b>Northern Europe:</b> Norway, Sweden, Finland, UK	<b>Norway, UK, Germany</b>
<b>East Asia:</b> China, Japan, South Korea	
<b>Africa:</b> Morocco, Tunisia, Algeria	
<b>N.A.:</b> Canada, Northern United States	<b>United States, Canada</b>
Australia and New Zealand	
<b>S.A.:</b> Argentina, Chile	

### 5.4 Where is MS found in the United States:



Multiple Sclerosis (MS) prevalence can vary significantly by region, but generally, higher rates are observed in areas farther from the equator. In the United States, certain states have been identified as having higher prevalence rates of MS. Some of these states include:

1. Minnesota: Known for having one of the highest rates of MS in the U.S.
2. Vermont: Also reports high MS prevalence.
3. New Hampshire: Similar to Vermont, New Hampshire has a higher-than-average rate of MS.
4. Maine: High prevalence of MS, consistent with trends in other northern states.
5. Wisconsin: Another state with a significant MS population.
6. Iowa: Reports higher rates of MS compared to the national average.
7. Washington: Has one of the higher prevalence rates on the West Coast.
8. Oregon: Similar to Washington, Oregon reports higher MS rates.

These states are generally located in the northern part of the United States, consistent with the global pattern where MS is more common in regions farther from the equator. However, it's important to note that MS prevalence data can vary over time and with new research, and access to healthcare services and diagnostic practices can also impact reported rates.

### 5.5 Vitamin D deficiency in the US:

Vitamin D deficiency in the United States is more common in certain regions and populations due to factors such as geographic location, climate, lifestyle, and demographics. The prevalence of Vitamin D deficiency tends to be higher in states with less sunlight exposure, especially during the winter months. Below are the regions and states generally known to have higher levels of Vitamin D deficiency:

Northeastern States	These states experience long, cold winters with limited sunlight, which reduces the natural production of Vitamin D through skin exposure to UVB rays.
Midwestern States	Like the Northeast, the Midwest has a significant period during the year when sunlight is scarce, leading to higher risks of Vitamin D deficiency.
Pacific Northwest	These states often have overcast weather and fewer sunny days, which can contribute to lower levels of Vitamin D.
Northern Plains States	The combination of high latitude and cold weather in these states can result in reduced sunlight exposure, leading to a higher prevalence of Vitamin D deficiency.
Alaska	Due to its extreme northern latitude, Alaska has very little sunlight for several months of the year, making Vitamin D deficiency more common.

#### Alignment of Regions/States with Vitamin D deficiency and High levels of MS

Regions/States with Vitamin D deficiency	Regions/States with High Levels of MS
4.1.1 Northeastern States <b>Maine, Vermont, New Hampshire</b> Massachusetts, New York	<b>Vermont, New Hampshire, Maine</b>
4.1.2 Midwestern States <b>Michigan, Wisconsin, Minnesota</b> Illinois, Ohio, Indiana	<b>Minnesota, Wisconsin, Iowa</b>
4.1.3 Pacific Northwest • Washington • Oregon	<b>Washington, Oregon</b>
4.1.4 Northern Plains States • North Dakota, South Dakota, Montana	
Alaska	

Table to create understanding of ideal human vs defect in human:

Ideal/Expected	Defect
Absorbs Vitamin D	Does not absorb Vitamin D from sun or diet
Immune system does not attack person	Immune system over protects, attacks the person.
Muscles grow and regenerate	Muscles degrade and not repaired
Vision is assisted with glasses, not pharma	Vision degrades and cannot be repaired
Walks with ease until a senior citizen or injury	Ability to walk becomes a struggle and only gets worse

#### 6 Create a Defect Report that pulls in the disease:

Date		Disease		Dataset	
Stage/Status		Duration		Severity	
Reproducibility		Expected Result		Actual Result	

<b>Environment</b>					
	Age	Gender	Diet	Geography	
Description					

## 7 Pulling it together: Bug -> Data -> Possible solutions

As I gathered data from publicly available datasets covering MS, I wanted to be able to isolate and develop a list of causations or early identifiers for MS.

- How to detect early?
- How to pause or stop progression?
- How to reverse the damage?

The list above was my ambition at the start of this citizen scientist project and paper. However, as I dove into the research of the disease using available tools, the need to make adjustments set in. My original goal was to use Chat GT4.0 to help me analyze large datasets, using my QA skillset I would look for specific patterns, assessing what is causes vs correlations, unfortunately, I hit a brick wall. I discovered that unlike software engineering, software development, medical research is a much tighter almost closed community.

I discovered that to get access to large data sets you had to be part of an organization- academic or research. Most of the data in Multiple Sclerosis out there is very tightly held. If I compare the medical research community to the software development, I feel spoiled. In Software we love thinking aloud, we push software programs in various stages – beta, beta 2, pre-production we are not afraid of failure. We are not afraid of someone “saying hey you did that wrong, I tested this and look I found these issues.”

Software development and Medical Research are two different industries, that is a given. However, one could easily say, software engineering is all about research and advancement. A major difference is our ability to embrace failure as part of the process of finding the answer.

With the adjustment made, I pivoted and became pretty good at prompt engineering. This still allowed me to use my QA skills to ask better questions. As a QA person, have a tendency to not trust everything I hear/read. I narrowed my toolset to ChatGPT 4 and Google AI Search Labs. I focused on addressing Multiple Sclerosis and the theory that vitamin D deficiency is a contributing factor to developing MS

Am I done? No, not at all. Here is my bug reply:

<b>Date</b>	9/1/2024	<b>Disease</b>	Multiple Sclerosis	<b>AI resources</b>	ChatGPT 4.0 Google AI
<b>Stage/Status</b>	Pre-	<b>Duration</b>	Unknown	<b>Severity</b>	HIGH
<b>Reproducibility</b>	.33% (1/300)	<b>Expected Result</b>	Person w/Vitamin deficiency <b>DOES</b> develop MS	<b>Actual Result</b>	Person w/Vitamin deficiency <b>DOES NOT</b> develop MS.
<b>Environment:</b>					
	<b>Age</b>	<b>Gender</b>	<b>Diet</b>	<b>Geography</b>	
	20 –40 yrs old	Female	Unknown	Pacific Northwest	
<b>Description</b>	<p>It is found that persons who develop MS also have a deficiency in Vitamin D. It is therefore theorized that if a person has low levels of Vitamin, they are at risk of developing Multiple Sclerosis.</p> <p>Based on an evaluation of the sources of Vit D – geography and diet. Initial focus on Geography, as it is a factor that is easier to evaluate on its own. Contrast to diet where there is great variety (vegetarian, vegan, pescatarian) in the US and the world.</p> <p>Findings: <b>More correlation over causation.</b> Looking at the common countries/regions from Chat vs Google, the Countries and Regions with high levels of Vit D deficiency are not nations/regions with high levels of MS.</p> <p>In the US, there is a greater, almost confirmable correlation that those who live in areas with high levels of Vit D deficiency and MS rates are in direct correlation, possible causation.</p>				
<b>Recommendation</b>	<p>Vitamin D deficiency occurs in everyone who is diagnosed with MS.</p> <p>Not everyone who has a Vitamin D deficiency develops MS.</p> <p><b>Based on publicly available data, the Vitamin D deficiency theory is at best a correlation, not a cause.</b></p> <p>If this were a software defect, I would remove Vitamin D deficiency from my list of possible cause and continue testing. I would look for a possible cause with a much greater reproducibility rate, .33% is not enough.</p>				

## 8 References

Lawlor, Megan. 2021 “1 in 400: How Many People Will Get ALS?,” ALS Therapy Development Institute, entry posted July 19, 2021, <https://www.als.net/news/1-in-400-how-many-people-will-get-als#:~:text=Blog%20in%20How,ALS%20Therapy%20Development%20Institute> , (accessed 6/29/2024)

Author and entry date not provided, “Luckiest Man”, National Baseball Hall of Fame, <https://baseballhall.org/discover-more/stories/baseball-history/lou-gehrig-luckiest-man> (accessed 6/29/2024)

O, Alona. 2023 “Software Quality Assurance, Quality Control, and Testing – the Basics of Software Quality Management,” TestFort, entry posted October 10, 2023 <https://testfort.com/blog/qa-qc-testing-the-basics-of-quality-management> (accessed 8/19/2024)

Parciak, Tina, et al, 2023. “Introducing a core dataset for real-world data in multiple sclerosis registries and cohorts: Recommendations from a global task force,” National Library of Medicine” entry posted December 23, 2023 <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10935622/> (accessed 3/3/2024)

Wallin, Mitchell, et al, 2019. “The prevalence of MS in the United States A population-based estimate using health claims data” entry posted March 5, 2019, <https://www.neurology.org/doi/10.1212/WNL.0000000000007035> (accessed 3/32024)

Collazo, Iris Marin. 2023 “ [Vitamin D and MS: Is there any connection?](https://www.mayoclinic.org/diseases-conditions/multiple-sclerosis/expert-answers/vitamin-d-and-ms/FAQ-20058258?p=1)” Mayo Clinic, entry posted April 19, 2023, <https://www.mayoclinic.org/diseases-conditions/multiple-sclerosis/expert-answers/vitamin-d-and-ms/FAQ-20058258?p=1> (accessed 4/17/2024)

Yadav, Yash, 2022. “Identifying plants and it’s medicinal properties., “ entry posted, posted 2022, <https://www.kaggle.com/search?q=Vitamin+D+deficiency> (accessed 6/20/2024)  
<https://medlineplus.gov/multiplesclerosis.html>

# Enhancing Validation Through Attestations

Brent Clausner

beclausner@sei.cmu.edu

## Abstract

Validation of software involves a variety of tests and processes to make it a robust product. Each step is essential to build a release. Utilizing attestations provide proof that the process was followed and that no unauthorized changes have been made throughout creating build artifacts. These attestations can be provided along with the software release package. Providing attestations and a plan layout to your customer gives them confidence that your organization has produced what is provided. Several checks and balances are included. This can eliminate the manual process of performing checksum examinations. This isn't an industry requirement yet and the responsibility lies with the person installing the software. An open-source project, like in-toto, can help to make the supply-chain of your product more secure. These steps can be added to your build pipeline to streamline the whole process. This paper will demonstrate how attestations can enforce policy and give consumers confidence that they have correct products.

## Biography

*Brent Clausner is a DevOps Engineer at The Software Engineering Institute. His background is rooted in software development having worked as a Software Engineer, Quality Assurance Engineer, and DevOps Engineer. He is a proponent to have projects adhere to the best practices and advocates for robust secure code being implemented.*

Copyright 2024 Carnegie Mellon University.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific entity, product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute nor of Carnegie Mellon University - Software Engineering Institute by any such named or represented entity.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License. Requests for permission for non-licensed uses should be directed to the Software Engineering Institute at [permission@sei.cmu.edu](mailto:permission@sei.cmu.edu).

DM24-0744

[Distribution Statement A] Approved for public release and unlimited distribution.

Excerpt from PNSQC Proceedings  
Copies may not be made or distributed for commercial use

PNSQC.ORG  
Page 2

## 1 Introduction

Typically, when someone purchases or licenses software for use, they download the release package, any binary executables or libraries, and should check the checksum hash against what is provided by the company you're purchasing or licensing from. This also includes any open-source software. This is meant to help ensure that you are getting what is expected. Without doing that you risk a man-in-the-middle attack. Including more protection for the customer in this case is always a good thing. If the packaged software includes multiple executables and reference files, doing this can be a daunting task. As an end user of any software, you should have a way to validate that the obtained software followed the providers expected process and you obtained the same released software.

Making use of attestations from your software suppliers can help to prevent attacks like what happened with Solar Winds. Using attestations goes beyond simple checksum verification as it is included in the attestation verification process. Providing attestations and layouts for your software product, helps to prevent man in the middle attacks and allows your customers to be sure they got what you intended.

The purpose of this document is to go over what processes can help make the customer aware that not only your build and testing process was followed, but also protecting internal code and output from commands from leaking to the public. We will be exploring the use of in-toto within an organization as a means of validating all testing steps. Doing this, will give the consumer of your product a way to have more trust in your software.

## 2 Attestations

A software attestation can be defined as, [1] “An authenticated statement (metadata) about a software artifact or collection of software artifacts”. These are small files that contain a lot of metadata.

### 2.1 Attestation Contents

Attestations can contain a variety of metadata about something in the build process. Often this includes who executed a command, what output the command produced, environment variables, the command return code, what files exist before the command ran, what files exist after a command ran, and all the sha256 checksum for every file.



*Figure 1: Envelope and contents*

[5] According to Supply chain Levels for Software Artifacts Committee (SLSA), an attestation must contain 3 things.

1. An envelope which contains a message containing the content in a statement and a signature.
2. A statement that binds the attestation to a set of artifacts. That statement is made up of a subject that identifies the artifact.
3. A predicate that contains metadata about the subject which normally contains a link.

## 2.2 Current Tooling

A few tools exist to allow generation and validation of attestations. This paper will focus on, In-toto. It is an open-source tool that allows you to generate and verify attestations. The commands can wrap whatever existing commands you have to produce those attestation files.

[2] Witness is an open-source tool that allows you to generate attestations, and can be used for verifying those attestations. It implements in-toto specification, ITE-5, ITE-6, and ITE-7. In-toto Enhancements (ITE) are used to track specific functionality for in-toto. These enhancements are specifications similar to a key to an issue tracking for in-toto.

[3] Another tool is a Google based Kubernetes solution called Binary Authorization. It can be used to verify images are signed by trusted authorities. This could be used along with in-toto.

### 2.2.1 In-toto terminology

This portion is important to describe terms used in this paper that are related to In-toto.

[4] **Step** is referenced as a single action that is signed by the functionary that executed it.

[4] **Inspection** is a client-side validation step. This gets included in a layout that can do additional validation outside of command file create/delete.

[4] **Layout** is a file that specifies each step necessary and who is supposed to execute the commands to create a final product. It also specifies what materials and products are expected at each step. Inspections are also able to be specified here as well to do additional validation.

[Distribution Statement A] Approved for public release and unlimited distribution.

[4] **Materials** are specified as files that are expected to be there at a step. These can be products from other steps. It consists of a list under the tag “expected\_materials”. Files are specified as a string and is a relative path to the file.

[4] **Products** are the files that are expected to be there because of a command. Materials can be included from other steps. It consists of a list under the tag “expected\_products”. Files are specified as a string and is a relative path to the file.

## 2.3 Layouts

The layout has all the steps necessary to create a final product. It's easiest to use a python file to generate the layout for you. Layouts can expire so it is useful to have it capable to be regenerated by having the layout defined in python. The layout is in JSON format. It being read in to the “*in\_toto.models.layout*” module. With python's use of whitespace this makes it easy to read though it can be very long.

It consists of the “payload” that is a generated string based off the python JSON input and signers' signature. The string is not human readable as it appears hashed off those two inputs. This helps in preventing bad actors from manipulating it and requires that the generator have private key access.

### 2.3.1 In-toto Layout Steps

The contents of the layout consist first of steps. The “steps” tag is a list that consists of several dictionaries. The steps define what files are allowed to exist in the “expected\_materials” tag. The files are specified as a string with a full relative path. Files listed outside of this JSON list within the layout will cause the verification to fail. Wildcards are acceptable if you have a source code directory that has many files to be made and removed. Similarly, with products of that step, you can specify the directory for build objects with a wild card for creation. This is done by using the “expected\_products” tag to list what files will exist and should exist after the command of that step is run. Files being created that are not in that list will cause the verification command to fail.

The command that the step is, is a list under the tag of “expected\_command”. For instance, if you were to want to build a rust project using cargo, normally the command is “cargo build”. With it being a list it is argument separated so it would be “[‘cargo’, ‘build’]”.

```
{
...
"steps": [
  { "name": "...",
    "expected_materials": [ ... ],
    "expected_products": [ ... ],
    "expected_command": [ ... ],
    "pubkeys": [ ... ],
  },
  ...
]
}
```

*Figure 2 JSON Sample of Steps in a Layout*

Both materials and products have similar syntax. They consist of a list and start with a keyword. The keyword determines what action has taken place to determine if a file is valid or not. Those keywords are:

- “ALLOW” is meant to indicate that a file is allowed to be there.
- “CREATE” is meant to indicate that a file is expected to be created because of the command. This is an “expected\_product” specific keyword.
- “MATCH” keyword allows you to match materials or products from other steps, even within the same step. For example, a product can match the materials from the same step.
- “DISALLOW” keyword is normally the last entry in the list. Typically is followed by the “\*” to have it error on any unexpected files that are created or are already there that aren’t intended to.

### 2.3.2 Private and public keys

Currently a few methods are supported, they include RSASSA-PSS, ED25519, and ECDSA. The private key is used for either signing the layout and/or signing the creation of the attestation. I recommend having a separate signature that’s used for signing the layout creation and use the public keys of user accounts that will execute the steps. Keeping the keys secure is the most important factor to using this tool. With the layout signer private key, a bad actor would have the ability to change what commands are run as part of the layout. Private keys must be guarded.

## 2.4 Attestation generation

In general, the command gets wrapped by “in-toto-run” command. Several things get passed into the wrapping command, such as the private key, name of the step, timeout, and finally the command to execute. The result will be that a file referenced as a link file will be created. This file will be named as step name, followed by a keyed prefix, and finally the extension of “link”. That “link” file is the attestation.

## 3 Intended Attestation Usage

The normal way an attestation is intended to be generated is part of the build process. Normally the build process consists of cloning of a repository, build/create the executable, and package the build artifacts. [Distribution Statement A] Approved for public release and unlimited distribution.

This process is to ensure that the end build executable is built from the correct source in the manner laid out by the layout. This is referred to what is listed under the “expected\_materials” tag. The result is all of the software’s package materials, attestations, and layout would be provided to the consumer of the software so they can verify it themselves. This provides evidence that the build/files all match that were used throughout the entire build. Nothing was modified when it wasn’t expected to.

With all the attestations and the build software, executing the command “in-toto-verify”, along with its command line arguments, “in-toto-verify” will verify all the attestation file contents with each step. This replaces checking the shasum against whatever is stated at the source of download. This method goes further into verifying all components in the entire process.

In-toto-verify will also run “inspection” commands that exist in the layout. The “inspection” is a list of commands that run and specifies what to check. For example, if your software compresses with tar, an inspection command could be to uncompress the file with untar. The “expected\_materials” for that inspection should contain the tar file matched to a “step” within the layout. The “expected\_products” tag should include the files matched with other steps in that layout for the tarred file.

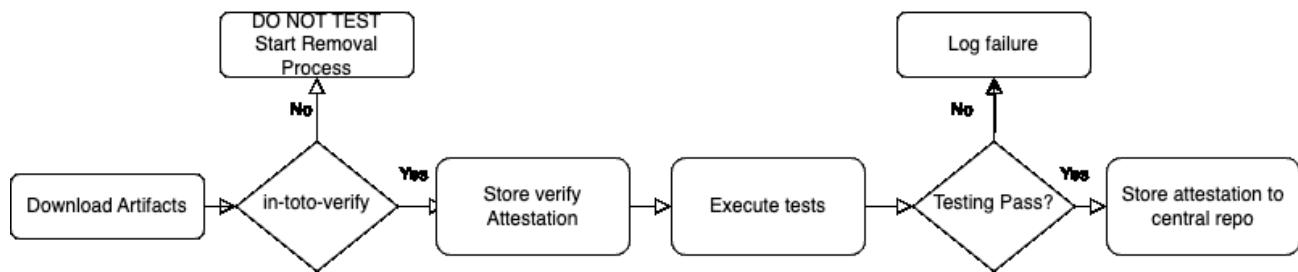
## 4 Proposed Attestation Usage

### 4.1 Internal Company Usage

Using attestations can be done with a pipeline to ensure that the build comes from where it’s expected to and built in the way dictated by the company that is producing the software. These can be packaged up and pushed to an artifact repository. The artifacts can be downloaded and verified that they match the build through the layout for that company.

When you execute commands there can be a lot of information in that command. The arguments to a build command might describe paths for files or even some sensitive information like passwords. Output from those commands can also show parts of source code, documentation, or again sensitive information like passwords. This information, if it is made public, can have a significant negative impact to a company.

What I’m proposing is that anyone that is doing testing on software that is through their build system, to verify the build artifacts with attestations and a layout as a first step. This attestation won’t be made public as only the verification command will be logged. From here any number of layouts can be used. As some manual testing inevitably will be done on the build, I propose to use a command that would be something like a command that doesn’t do anything, like “echo PASS” to indicate that testing has passed. That command will just output the string “PASS”. If you have multiple teams, a final layout can be used that contains checks for an attestation created by those teams and signed by a team lead or program manager.



*Figure 3 Proposed flowchart of actions*

With multiple teams the flowchart above can be followed. You should be storing of all the attestations onto a central server or repository when all tests pass. The only attestations that will be provided to any

[Distribution Statement A] Approved for public release and unlimited distribution.

consumers will be the ones that issue “in-toto-verify”. This will prevent giving internal build output to end users, though they can verify that the build has been carried through the rigors of the testing process and has been delivered to the end user by the verification of the attestations with “in-toto-verify”.

This also provides the company a trail of who did what to ensure quality in the software product. These attestations give proof that someone is signing a file that testing has been done.

#### **4.1.1 Hypothetical outcome**

If you are working for a company that for instance would create an operating system, your organization is most likely going to have a lot of different functional teams. If something happens where a function of that operating system does not work as expected when saving files to disk and that bug is found after the product is released by a customer, this can give the company visibility into where the gap is.

With attestations, it allows the user to verify the exact version is from the expected manufacturer. It can help to prevent man in the middle attacks. This also allows the end user the capability to provide the manufacturer those attestations when there is a problem such as a claim above. They will most likely request other additional information such as log files. The exact version could be determined from that content and that it is in fact the manufacturers product.

A claim that is found without this type of protection can lead to a false line of validation. Without having a manner to determine the exact version of software that a user is using and that it hasn't been tampered with, this wouldn't be worth investigating. This could happen if it was downloaded from a third party website. If there are not attestations, layout, and public keys, it makes it difficult to truly verify that the software is from the expected manufacturer.

## **4.2 Quality Assurance Testing**

Using multiple layouts from within a company can help to ensure the necessary process is followed. For instance, including known processes either via a pipeline or manually of doing static code analysis is a good thing to do. The output and command arguments may not be something that you would want consumers to have access to.

There may be things in code analysis, such as a static code analysis “lint” check, that you are ok with doing that tools pick up as a violation. Something like white spacing can often show up. When that does it can include lines of code in the output. The output of the command is in inside the attestation and that could lead to leaking the information outside the organization if are shared.

#### **4.2.1 Multiple Layouts**

Using a layout that is from a build process to verify the version and executable is necessary as a first step for anyone doing testing. The verification step of the build should have an attestation created for that. That should be the first layout that would be in use.

The following JSON is an example for the first layout. You can see where files could be, like under “src/\*” and what commands are used to build, “cargo build”.

```
layout = Layout.read({
    "_type": "layout",
    "keys": {
        key_owner["keyid"] : key_owner,
    },
    "steps": [
        {
            "cmd": "cargo build --release"
        }
    ]
})
```

[Distribution Statement A] Approved for public release and unlimited distribution.

```

    "name": "building",
    "expected_materials": [
        ["ALLOW", "src/*"], ["ALLOW", "*.link"], ["ALLOW", "*.pub"],
        ["ALLOW", "*.layout"], ["ALLOW", "Cargo.toml"]
    ],
    "expected_products": [
        ["MATCH", "*", "WITH", "MATERIALS", "FROM", "building"],
        ["CREATE", "target/*"], ["CREATE", "Cargo.lock"],
        ["DISALLOW", "*"]
    ],
    "expected_command": [
        "cargo", "build"
    ],
    "threshold": 1,
    "pubkeys": [key_owner["keyid"]]
},
{
    "name": "lint_check",
    "expected_materials": [
        ["MATCH", "*", "WITH", "MATERIALS", "FROM", "building"],
        ["MATCH", "*", "WITH", "PRODUCTS", "FROM", "building"],
        ["DISALLOW", "*"]
    ],
    "expected_products": [
        ["MATCH", "*", "WITH", "MATERIALS", "FROM", "building"],
        ["MATCH", "*", "WITH", "PRODUCTS", "FROM", "building"],
        ["DISALLOW", "*"]
    ],
    "expected_command": [
        "cargo", "clippy"
    ],
    "threshold": 1,
    "pubkeys": [key_owner["keyid"]]
},
{
    "name": "package",
    "expected_materials": [
        ["MATCH", "*", "WITH", "MATERIALS", "FROM", "building"],
        ["MATCH", "*", "WITH", "PRODUCTS", "FROM", "building"],
        ["DISALLOW", "*"]
    ],
    "expected_products": [
        ["MATCH", "*", "WITH", "MATERIALS", "FROM", "building"],
        ["MATCH", "*", "WITH", "PRODUCTS", "FROM", "lint_check"],
        ["CREATE", "package.tar"],
        ["DISALLOW", "*"]
    ],
    "expected_command": [
        "tar", "-cf", "package.tar", "target/debug/simple",
    ],
    "threshold": 1,
    "pubkeys": [key_owner["keyid"]]
},
]

```

[Distribution Statement A] Approved for public release and unlimited distribution.

```
"inspect": [],
})
```

The layout above is used with creating and verifying the build. The verification of that layout and attestations should be the first step in doing manual testing. The ‘in-toto-verify’ should be wrapped in the ‘in-toto-run’ command to generate the attestation. After all the manual testing is completed, if all tests passed, executing the command “echo PASS” wrapped by the ‘in-toto-run’ command. This will generate an attestation that can be provided to a release manager, who can run the last verification command.

Having a layout for a step prior to release should include verifying that each team did their own verification of the build components, executed tests with an attestation from a designated lead for that team, and updated test tracking software. This would be a second layout that can be used. At this point, the layout, attestations, and public keys could be provided to the public for verification, but that can be a lot of files and additional space being required wherever the release files are stored.

```
layout = Layout.read({
    "_type": "Layout",
    "keys": {key_owner["keyid"]る: key_owner},
    "steps": [
        {
            "name": "verify",
            "expected_materials": [
                ["ALLOW", "in-toto-files.tar"], ["ALLOW", "*.link"], ["ALLOW", "*.pub"],
                ["ALLOW", "package.tar"], ["ALLOW", "*.layout"],
                ["ALLOW", "target/debug/simple"], ["DISALLOW", "*"]
            ],
            "expected_products": [
                ["MATCH", "*", "WITH", "MATERIALS", "FROM", "verify"],
                ["DISALLOW", "*"]
            ],
            "expected_command": [
                "in-toto-verify", "-l", "./build.layout",
                "--verification-keys", "./owner.pub", "--link-dir", "./"
            ],
            "threshold": 1,
            "pubkeys": [key_owner["keyid"]]
        },
        {
            "name": "testing",
            "expected_materials": [
                ["MATCH", "*", "WITH", "MATERIALS", "FROM", "verify"],
                ["DISALLOW", "*"]
            ],
            "expected_products": [
                ["MATCH", "*", "WITH", "MATERIALS", "FROM", "verify"],
                ["DISALLOW", "*"]
            ],
            "expected_command": [
                "echo", "PASS"
            ],
            "threshold": 1,
        }
    ]
})
```

[Distribution Statement A] Approved for public release and unlimited distribution.

```

        "pubkeys": [key_owner["keyid"]],
    }
],
"inspect": []
}
)
```

The above layout will validate that the build is what's expected and can be expanded for multiple teams with the step “testing” above. Additional steps can be added there that would signify the functional teams and system testing teams that an organization might be using. The verification of this should be used as the last attestation that would be public facing. The output from this command will only show any information from the ‘in-toto-verify’ command. This will not include any output of actual code from the build process.

Example layout that can be used for release:

```

layout = Layout.read({
    "_type": "layout",
    "keys": {key_owner["keyid"]: key_owner},
    "steps": [
        {
            "name": "release",
            "expected_materials": [
                ["ALLOW", "package.tar"], ["ALLOW", "*.link"],
                ["ALLOW", "*.pub"],
                ["ALLOW", "*.layout"], ["ALLOW", "target/debug/simple"],
                ["DISALLOW", "*"]
            ],
            "expected_products": [
                ["MATCH", "*", "WITH", "MATERIALS", "FROM", "verify2"],
                ["ALLOW", "*.layout"], ["ALLOW", "target/debug/simple"],
                ["DISALLOW", "*"]
            ],
            "expected_command": [
                "in-toto-verify", "-l", "./internal.layout",
                "--verification-keys", "./owner.pub", "--link-dir", "./"
            ],
            "threshold": 1,
            "pubkeys": [key_owner["keyid"]]
        }
    ],
    "inspect": []
})
```

Having a simpler layout as a final version for release, as shown above, should include the verify step to check the layout, before release. The output for the attestation produced for public release would only include the command, under “expected\_command”, to verify the testing layouts. The included file metadata from this would be the files that are for release have matched to the ones in all the layouts previous.

The examples shown above is an extremely simplified version of what I'm proposing. Additional test steps could be added to the first layout for quality assurance pieces or additional layouts could be added, with [Distribution Statement A] Approved for public release and unlimited distribution.

multiple verify steps within second layout before the release layout. There are a lot of ways to handle this. The sample above using three layouts does work. The second layout that includes a verify of the “build.layout” of the first would be a good example of what any test engineer could run prior to capturing tests. The release manager should still have a small layout at the end that has a layout that includes all the test attestation and verifies it. That verify attestation would be what’s provided to a customer.

### 4.3 Customer Deliverables

The following should be provided to customers:

- Final verify attestation
- Release layout
- Public keys
- Release software

If the release attestation is generated in a folder that only includes the links and release materials, the link files that were used for the verify step within will be excluded from what the consumer will receive. This means the filenames and output from commands will not be shipped to them, just the materials necessary to run the software and the files for verification.

## 5 Conclusion

There is an initial up front cost for setting this up. Having someone to coordinate the layouts and maintain them is a cost. Once they are created, maintaining them isn’t as difficult as they can be modified and regenerated on demand. The software release manager should oversee the final layout. They will be required to work with leads from functional teams to coordinate the attestations from them and the expected commands and layouts in use for the various teams.

The final layout that would be provided to customers should be small and include a single “in-toto-verify” command. This reduces the output captured for the attestation and ensures that the entire build/release chain has been validated the whole way through.

### 5.1 Customer Impact

Having all these together allow a customer to verify everything occurred from the perspective of the manufacturer. If there ends up being major complications for what the software was meant to do, it can be traced all the way back to the test team. As long as the manufacturer has stored testing logs, it can be seen if there were any mistakes made in the testing process. This holds the manufacturer accountable for what they’re doing.

This should give customers more trust in the manufacturer. One thing to know about downloading software is that most companies provide some hash shasum that can be used to verify the downloaded object is the expected version. Using the in-toto verification adds to this process and checks all of those for you, so it is very straightforward for a customer to verify the release software.

## References

- [1] SLSA Committee, "SLSA \* Terminology", Supply chain Levels for Software Artifacts, <https://slsa.dev/spec/v1.0/terminology> (accessed May 6, 2024)
- [2] The Witness Contributors, "Witness | Witness In-toto", Witness.dev, <https://witness.dev/docs/> (accessed May 6, 2024)
- [3] "Binary Authorization | Google Cloud", Google Inc., <https://cloud.google.com/binary-authorization> (accessed May 6, 2024)
- [4] "in-toto-spec.md", In-toto, <https://github.com/in-toto/docs/blob/v1.0/in-toto-spec.md#17-terminology> (accessed May 6, 2024)
- [5] SLSA Committee, "SLSA \* Attestations", Supply chain Levels for Software Artifacts, <https://slsa.dev/attestation-model> (accessed May 8, 2024)

Figure 1: Licensed from Adobe Stock images and modified for this paper.

Figure 2: Produced by Brent Clausner

Figure 3: Produced by Brent Clausner

# Greater Quality Through Tactical Domain-Driven Design

**John R. Connolly**

john@articulatedomain.com

## 1. Abstract

Quality software, for many, means that software does the work it needs to do today without any defects, i.e. functional quality. This narrow definition gives us a sense of success in the present, but what about the software's ability to absorb change for the future? The reality is that for many organizations millions of dollars of maintenance budget are wasted on software that functions well for the initial intent but does not absorb new features easily. Newer measures in the DevOps arena such as the DevOps Research and Analysis (DORA) metrics are helpful when learning how adaptive we are. But how do we improve those numbers and potentially decrease total cost of ownership when change is impacting the business positively and the software change is not able to keep up? Today, in addition to all the attributes of quality software we have come to expect, we now must now consider at least one non-functional quality attribute. That attribute is *adaptive* or *adaptability*. This is important because useful software seems to always need new functionality. Many have experienced great pain of trying to update software not designed to absorb change. Domain-Driven Design (DDD) and related paradigms, tools, and mindsets can help us get to new levels of adaptability. Specifically, Tactical DDD enables teams to set and respect software boundaries based on the domain the software it serves. These clear boundaries make code much more navigable and change far less frustrating. Tactical DDD then can be augmented by other paradigms like advanced event driven design, promoting even more autonomy of function reducing the ripple effects of change. This can create new levels of non-functional quality that typically reduces wasted maintenance budget and improves the performance of the software development life cycle (SDLC) for all who contribute to the software product.

## 2. Biography

*John R. Connolly started his career in database development in 1997 in the U.S. Air Force and has a B.S. in Organizational Management. He is currently the Principal Consultant of Articulate Domain, a company he started in 2019. He specializes in aligning solution designs with the problem domain through Domain-Driven Design, distributed systems design and uses tools like Eventstorming (Brandolini) to drive collaborative models with business, stakeholders and developers. John holds a patent for systems design of essential oil distillation automation. John is completing a master's degree in the clinical mental health field and resides in Salem with his wife Sandy and two cats.*

Copyright John Connolly 2024

# 1 Introduction

## 1.1 Current State of Software Design

Logical code can easily get disorganized fast as organizations ask developers to apply new feature after new feature quickly. In the industry, we have a term for code that is generally hard to navigate and reason about as a whole. We call this a Big Ball of Mud (BBoM). BBoMs are usually produced in what is often called a “Feature Factory” where backlogs are processed day in and day out. These feature factories generally do not have a *behavioral* design paradigm with which they ascribe or utilize.

The BBoM architectural model is so popular that it is now considered by many to be the most widely adopted architectural pattern in the world. It's not intentional, but it is pervasive. Many believe it is inevitable that software eventually just must be difficult to advance with new capabilities. This perpetuates the adaptability quality problem. BBoMs are so difficult to work our way out of that many tend to acquiesce to the enhancement delays they cause. Once it gets too difficult to maintain, most opt for a rewrite of the whole system or choose to never upgrade the software again in any significant way, losing great competitive advantages. With the advent of new design paradigms like DDD there is not only a way to avoid these conditions, but there are paths to repair the BBoM as well.

## 1.2 Low Quality Costs

The total cost of ownership of maladaptive software can potentially ruin a company's bottom line. For those who need empirical data to better understand this problem can read the robust book on the topic by Capers Jones and his team, “The Economics of Software Quality”, where he analyzed hundreds of companies and came to the conclusion that “Although formal design and code inspections originated more than 35 years ago (as of 2011), they still are the top-ranked methodologies in terms of defect removal efficiency.” (Jones et. al. 2011)

The waste in 2020 and 2021 was not insignificant. “According to ‘The Cost of Poor Software Quality in the U.S.’ by the Consortium for Information and Software Quality (CISQ), the collective bill in the U.S. for defective software in 2021 was an estimated \$2.41 trillion, up almost 16% from the 2020s \$2.08 trillion. That’s more than the GDP of all but a dozen countries. And it doesn’t even count an estimated \$1.52 trillion in ‘technical debt’ (TD)—accumulated software vulnerabilities in applications, networks, and systems that have never been addressed but will have to be paid eventually.” (Poor Software Quality Can Cost Time and Money, Straightforward Solutions Are Available 2023)

## 1.3 Didn't Agile Solve This?

Agile was introduced to the market in 2001 to course correct the hyper extended delays of the Waterfall SDLC model (“Manifesto for Agile Software Development.” 2001).

There is a very good reason why Agile did not solve this adaptability issue. The word Agile was meant to convey the idea that *projects, teams and software* could be more agile. In practice, however, the industry focused primarily on agile projects and teams rather than systems themselves (Agile and the Long Crisis of Software n.d.). This lulled many leaders to implement Agile as a team management practice in a sort of assembly line type of mentality. While this team-oriented focus provided iterative *project management* alternatives to the long and drawn-out phases of waterfall, it generally did not address the need to design novel systems in any significant way. These project management activities became a commercial opportunity cementing these inefficiencies into contracts with companies providing Agile frameworks, often called Scrum, SAFe etc. Even with the advent of product-centric thinking, the industry did not address these design needs as there was more of a focus on task flow than design guidance in many of those implementations. Agile implementations mainly focused on project process and ceremony with little to no attention to curbing the tide on poor designs. As management pushed developers to produce new

features faster, measuring them by their sprint goals rather than their solution outcomes, many felt they needed to cut corners leaving no room to design adaptive systems.

## 1.4 Wait. Waterfall Again?

It may seem that introducing this idea of behavioral design could then take us a step back in time to the days of Waterfall. Waterfall had introduced a lot of rigid steps that prevented any change that would require rolling back to an already completed stage of the SDLC. This forced most projects to keep any stage of the process open for a long time to ensure no changes would be asked for before moving on to the next stage, creating a slow overall delivery model (Abraham 2022). Design had to be fully complete before development could start. Development had to be fully complete before testing could start and so forth. In addition, when a project was started and completed, the documentation had to be accurate and complete for all the capability captured. The waterfall process was so arduous that modifications were generally too expensive to accomplish. While Waterfall did have a lengthy design step, that level of design and detail was so extreme that software rarely made it to market in time to meet the demand. That is why Agile became so popular. Waterfall design is often referred to Big Up Front Design. Agile for the most part never implemented an intentional design step sponsored by the domain. This all or nothing approach to behavioral design is extreme on both ends and does not strike a good balance. Iterative behavioral design should be integrated into the SDLC.

## 1.5 Becoming Adaptive Intentionally

We will utilize the term *adaptability* rather than agility to reference our new key quality software design attribute to reduce confusion. In the long run, the quality of being adaptive, or software adaptability, can provide companies with something they often run out of - time. We already know that, while design adds energy to the SDLC, most CTOs know it generally saves time during the UX development cycle. In fact, good UX design often subtracts workload at the UI developer's keyboard. DDD translates that same benefit into the logical layer of software. Utilizing Tactical DDD principles, developers and testers alike have a much clearer target and a workable vision. When they know where methods and their related properties should coalesce and where they should not, this can reduce churn significantly. Therefore, it can take less time to develop software to a mature adaptive state. If teams want to serve their companies well for the duration of the need, which is often years, it will pay to adopt principles of DDD. This can be iteratively applied to the SDLC and will help teams steer clear of the BBoM outcomes.

DDD addresses two primary software design concerns that can impact the SDLC. They are Strategic and Tactical (Evans 2003). We won't spend much time on the Strategic concepts of DDD except to say that Domain-Driven Designers work on strategic designs when learning how to draw the line between one subsystem and another. Strategic DDD proposes investigating the notions of Subdomains and Bounded Contexts to support those decisions. Once those boundaries are formed, then Tactical DDD can be applied inside those boundaries. Tactical DDD domain models can struggle internally which is a good thing to be aware of. That condition often highlights a need to reassess Strategic DDD boundaries. When this happens in design time, it saves incredible amounts of wasted energy when those interoperability choices only need to be altered on paper.

So, there is an interplay that often happens between Strategic and Tactical DDD focal points as each informs the other of clues to improve design choices at a macro and micro level. Tactical DDD provides for the protection of logic, cohesion of domain concepts and reliability of functionality within those boundaries. Tactical DDD aims to improve feature capability absorption while reducing side effects. Adaptive design tactically is a critical outcome of quality DDD.

## 1.6 Design is Already Socially and Culturally Accepted

As mentioned previously, UX design is a known improvement to the SDLC and is often required before front-end developers can craft User Interfaces. As soon as a new project is announced, one of the very first things that is accomplished is the UX department's design of the UI. Wireframes are mulled over and

refined. Time is spent gathering customer feedback and design opinions from UX experts to craft a rough draft of exactly how users should interact with the system and how that system should provide feedback to the users. None of this is generally programmed at the beginning. This is smart. It is smart because it reduces waste when the system is in development mode. UI developers will have far fewer questions while crafting real-world applications at the presentation level because the context is made very clear by the design guides and the wireframes that show what is needed. As we will discuss, the same is true for behavioral design via Tactical DDD and is further economized by the tools behavioral designers are using to accomplish these designs.

To put a finer point on this correlation between UX and behavioral design, UX Design is desired because it makes development more profitable in several ways. Customers more readily like the first truly developed representation of the systems under development and meet the acceptance criteria more quickly. Developers do not have to spin code like a never-ending unsolved Rubik's Cube. A quality, well thought-out, UX design infuses a sense of direction and confidence into the process.

The great thing about the UX design model is that it can be used for future changes. This level of iterative design is also a benefit for the developers and the domain as whole as it saves time and keeps the design domain appropriate.

## 1.7 But We Make Data Models!

Data is not behavior, yet it is highly affected by behavior. Many IT leaders push for data models on the heels of having a good UX design or even in parallel. They have been trained through decades of *application development* reinforcement that teams need to connect the UX design to data models well to have a great system. If it were only so.

Udi Dahan, who masterfully teaches an Advanced Distributed Systems Design course, rightly points out that most development leaders and architects were trained in the days of application development. The world shifted to API and Event-Driven systems with decoupled modules or microservices, but the developer base was never fully trained on what that looks like when designing *systems* of logical software behavior. Decades ago, applications were singular systems with one database, the entire UI and all the behavior. Access 2, FoxPro, and many other desktop software development packages empowered software crafters to do it all in the 1980's and 90's. UI, Business Rules and Data coexisted in the same file. When servers came around, that development paradigm was applied to distributed systems. Distributed systems need an added layer of intentionality if they are going to be designed to adapt to new capabilities in the future.

# 2 The Ultimate Heart of the Matter

## 2.1 The Missing Piece

Intentional domain honoring design is generally not well understood and is not implemented to make behavior reduce friction for the business it serves. If this were a tiny fraction of the software developed, this paper would not be written. Often, clients of Articulate Domain, are asked, "Percentagewise, how much of the code in your more mature systems are the business rules or the behavior of the system vs the presentation and data definitions?" Invariably the answer so far has always been between 65-80% behavior. If true for most complex systems, then this means that most teams are leveraging the power of quality and time-saving design for only 20-35% of the mission critical software they are developing. Most of the companies interviewed have never heard of DDD or behavioral design in general. Hopefully by now we have surfaced a plausible reason why many quality issues, adaptive or otherwise, might easily creep into any system.

It is important to note that all software is designed and does what it is designed to do. If behavior is not intentionally designed, then it is unintentionally designed as the developer codes the software. This is painful for software developers who are guessing what a good domain honoring design would look like. If

companies won't make UI developers program without good wireframe design, why do they ask or even tell the business logic developer to churn through poor representations of behavior until one day they present a working system? And, once those systems are more mature, they are so maladaptive to change, developers will refuse to modify these rigid systems in any significant way. To make matters worse, it seems to be that many developers quit this type of job to get hired at another company where they will likely and unintentionally build another maladaptive system.

## 2.2 Testing Matters Too

It is also important to note that designing the behavior of systems is directly related yet not the same as designing tests. Tests should be designed and developed in the context of the behavior of a system. It is hard to test a system that is designed by developers on the fly, better known as emergent design, because the target design changes randomly as the developers realize their designs do not quite meet the need. If we are following a DDD paradigm then the design should point very easily to the need of the domain. These designs will feed into testing initiatives so that testing can be more confident as well. This includes Test-Driven Development (TDD) and Behavioral-Driven Development and other testing paradigms. Once domain needs are identified and are agreed upon, then tests at any level should reflect the needs at the level of the test. For instance, a unit test should ensure not only that the method works but works according to the domain. Any threshold, or limitation the domain puts on that method should be included and so forth.

In a distributed system, the domain need is generally supposed to be divided up into smaller services that perform specified capabilities. In this case, sub-system tests not only should be exclusive to the sub-system but should reflect any domain capabilities the system provides. When testing interactions between systems, generally governed by what we commonly call an *interface contract*, then those contracts should ensure domain concepts that impact those contracts are accounted for.

## 2.3 Little to No Intentional Design Can Work - Sometimes

For some software there is little need for Tactical DDD. Software that does not have a lot of behavior is a great candidate for such a decision. Just be sure there is not much behavior to plan for when making that design choice. It might be safe to design for a little bit to verify that it is unnecessary.

Another area that will not benefit much from a holistic design is the proof of concept. Unless the design itself is the focus of the investigation, the team should skip to just trying ideas that solve a specified problem. This type of project rarely has the adaptive rigor needed to ship products to production.

## 2.4 Some Software Benefits from Meticulous Behavioral Design

There is a time to do a lot of up-front intentional design for a longer period. This is a good thing for software that puts lives at risk or when profitability requires intricate clean design. Space modules and life-support systems are some examples of these devices that generally should not skip design steps and should be reviewed and tested for thoroughness and accuracy.

## 2.5 Most Software Needs Good Enough Design

Good enough design for complex software is hard to define but we do need a working definition. Good enough design is a process that affords a team the opportunity to model a solution that solves today's needs well and leaves that solution open to enhancement. This model should be accomplished quickly, answering critical questions well enough providing avenues for upgrades in the future. This is a skill that may take the proverbial 10,000 hours for a designer to reach an expert level of ability. Teams can often accomplish good enough design by taking time to decouple concerns at the transactional level quickly and making good domain honoring design choices.

## 3 Modern Behavioral Design

### 3.1 What is DDD Basically?

Eric Evans published his seminal book Domain-Driven Design: Tackling Complexity at the Heart of Software in 2003. A small segment of the industry knows about DDD, its uses, and benefits. Even though Microsoft consultants have written about it, European countries have generated a large following, Martin Fowler has described it, a plethora of videos are posted on YouTube, and Oxford is now teaching it as a software design course, it remains out of view of many.

Evans took programming practices up until the time of his book, and advanced abstract organizational thought to improve not only code structure but its relevance to the domain it serves. Object Orientation (OO) by that time was very much becoming the popular way to craft code, but it lacked a well-accepted paradigm that classified user requirements concepts as *domain* object types using *domain terminology* for its naming convention. OO also did not have an industry-wide method to craft *domain boundaries* based on the Language they use. Evans created a way to bridge this gap to make software much easier to work with by facing the complexity head on. DDD ended up being a language and technology agnostic paradigm that could be used by developers all over the world.

At a high level, DDD is a way to classify your domain for the sake of both the business experts and the IT experts alike such that there is no need to translate the terminology used in code into the terminology used in the business. There is one language per context and per context the business and IT speak the same language. In practice this is difficult to achieve, but the more it is accomplished the less friction there is in communication between business and IT. In fact, it is considered a desired outcome to make software look like something the business can understand no matter the software language used.

#### 3.1.1 DDD: A Closer Look

DDD at its core is what the name indicates. A more descriptive way to state it would be to say that DDD is a set of principles where Domain-Driven Designers can work with Domain Experts (or Subject Matter Experts) to map Domain Boundaries and Domain Capabilities into Domain Models that programmers should be able to clearly understand and can more confidently express in code. This may be hard to absorb, so let's look at it from other angles before we describe the programmatic elements of DDD.

The *domain* is the area of a business, agency or organization that needs some solution or set of solutions to solve problems, accomplish something or somehow advance the product or service offering. Many will label the domain as the *problem space*. Domains can be further subdivided into subdomains. Subdomains can be further subdivided into smaller subdomains.

High level domain examples are insurance, ecommerce, banking, healthcare, air travel, hospitality, gaming, retail, education and the list goes on. Those domains are supported by other subdomains like accounting, shipping and operations and those can be further subdivided. If DDD were easy, we would just name the organizations' various departments and develop a software system for each department with interconnected data. Those who have tried to do this generally and quickly run into a myriad of complications. While it is not helpful to naively use these subdomains in this manner, these areas of the domain own the ground truths essential to making DDD adaptive systems. It is important to explore these domain areas and take steps to design models that are inspired by them and support their mission well. When those domains are driving the solution designs, domain to system solution alignment increases significantly. Honoring the domain this way is a necessary ingredient for most complex systems to be highly adaptive. One important point to note is that DDD is a paradigm of design, and not a silver bullet. No two DDD projects are the same, yet they use the same basic principles to craft quality designs.

### 3.1.2 Tactical Domain-Driven Design Benefits

DDD tactically affords developers a way to encapsulate domain capabilities more cleanly, reliably and less coupled than they would without this design paradigm. Creating solutions this way removes many of the negative side-effects that stem from common software changes. Also, once this skill is embedded in each team member working on a DDD designed system, this paradigm gives those team members a more defined language to discuss the elements in the system. Knowing Tactical DDD when working with a system that uses this paradigm makes it easier to navigate lines of code. When all teams operate this way in a large organization, it makes it easier for a developer to move from team to team and still be productive. Additionally, if developers veer from the normalcy of their DDD coding practice, they can steer the course back or at least make a case for the anomaly. Being able to come back months or years after working on a codebase and confidently understand it is a great benefit of Tactical DDD.

## 3.2 Key DDD Tactical Elements

### 3.2.1 Involve the Domain Experts and Discover Their Language

Before we get into these important Tactical DDD building blocks it is critical to understand that using Tactical DDD building blocks well does not make software adaptive by itself. Using Tactical DDD to craft a rich domain map that serves the domain well for the future is the holy grail of DDD. We cannot just build highly organized code and call it good. ***We must organize these building blocks in a way that truly expresses and serves the behavioral needs of the domain it serves.*** This is the essence of DDD. If we do not use these building blocks to express and serve the behavioral needs of the domain, we might as well stop pretending to serve the domain with DDD. This does not mean that we achieve perfection in our designs. We should strive to understand the domain need and map solutions that are likely to adapt as those needs change. Domain needs are best described by the Domain Experts in the industry that is being served.

Domain Experts often need encouragement, empathy and inspiration to detail domain needs. This can be a process of learning, doing, and learning some more. This makes iterative Tactical DDD a fit in an Agile environment. This helps fill the void that an Agile implementation focused on projects (or products) and teams primarily suffer from.

A word of caution should be expressed here. Domain Experts can be influenced by the systems that they work in. In fact, clear, complete and concise domain term definitions often get replaced by definitions in antiquated software or OTS packages. This can clutter the conversation and is something a good facilitator of DDD modeling sessions will catch and explore. Language is important in DDD and making it clear is a key part of the paradigm.

### 3.2.2 Domain Contextual Language (Ubiquitous Language)

Communication always rides on the common understanding of a language. Morse Code, Egyptian Hieroglyphics, French - it does not matter. Even though there is a common understanding in the dictionary of the terms we use, often there are several nuanced definitions. This can wreak havoc when trying to understand each other clearly. Fortunately, when both sides are clear on the terminology, they communicate so well it is as if they can finish each other's sentences, or code as it were. When there is ambiguity or differences of understanding, assumptions and miscommunication make agreements weak or not even possible. How many times do we encounter missteps that started with multiple definitions for a word or multiple words for a definition. A DDD focus will clear up these issues.

Clear language is at the heart of DDD. Strategic DDD efforts push for clarity on what Eric Evans calls the Ubiquitous Language. Though, ironically, that phrase is confusing to most when they first encounter it. To clear up this term, what Evans is driving at is that within each contextual sphere of influence, the same words always have the same definitions. This does not mean that the same word must have the same definition in separate contexts. For instance, the word "invoice" might have three different definitions in three subdomain contexts within a single business context, but it should never have two or more

definitions within one context. This led to the term “bounded context”, indicating that terms and their definitions are bound and never used differently unless the domain changes the term or its definition. The Ubiquitous Language is likely easier to understand as a ***contextual language***. So, we will use that term in the sections that follow. Bounded contexts or the contextual language of a domain or subdomain is a work in progress. As the work to define these terms matures, so will the language. This is no different than any language that is long lived and evolving.

Once we understand that a bounded context is a boundary around contextual terms and their definitions, i.e. the contextual language, then many will ask, “where do we use those terms?” These terms are used throughout the culture of the domain. The business, agency, organization, their print material, their code base and any other format that uses these ideas should use the same language. In code, the nouns, verbs, adjectives etc., from the domain get used in the namespaces, properties, packages, modules, functions, methods, subroutines, interfaces, tests other constructs.

Gone should be the days of naming class objects or functions using the animal kingdom, favorite automobiles/motorcycles or sci-fi movie characters. ***Professional software should use the terms of the profession it serves***. This is critical to adaptivity because any translation between the domain and the software is one added point of potential communication failure. The more we remove translation need, the more likely we understand how to read the code against the backdrop of the domain.

### 3.2.3 Aggregates

When designing software from a Tactical DDD perspective, behavioral code is encapsulated in Aggregates. The systems outside of the aggregate communicate with the aggregate through an Aggregate Root. Inside the aggregate there will usually be domain behavior and information. These behaviors and information are organized into interconnected Entities, Value Objects, Domain Services, and Domain Events. New aggregates are often created by a Factory. Once the first successful execution of an aggregate is completed, the state is saved through a Repository into some datastore. Some will store the latest full state only, while others will save a new event on top of all historical events of that aggregate so that the state changes can be replayed later. Saving Domain Events in a series is known as EventSourcing – a topic we will not be expounding upon in this paper. Some of these terms may not be clear since these terms can be used in an overloaded fashion.

**3.2.4 An aggregate is a conceptual whole of a capability with all the behavior and data needed to achieve that capability. Aggregates take a command to process domain behavior and if needed store the resulting data. After the Aggregate attempts to fulfill that command, whether it succeeds or fails, it reports the appropriate outcome back to that calling party. The critical thing about this is that no caller can directly call an element inside the aggregate. They can call only the root of the aggregate. All implementation details are hidden from view from all other systems.**

The aggregate root is generally an entity, but not always. The key point is, there is one point of contact on the aggregate called the aggregate root and it exposes the interface(s) that afford other systems the opportunity to request that the aggregate attempt to do what that aggregate was designed to do.

Aggregates enable adaptivity in that if the behavior can be updated and the name of the commands are still relevant, then we do not have to make changes to the rest of the connecting systems. Also, if the aggregate can be extended without harm to past transactions, then we are backwards compatible as well. If the aggregate is not capable of being extended, then we can make a new aggregate with new functionality like versioning an API. This can become cluttered with too many aggregates, but the design concept forces us to think deliberately about this choice.

A real-world example of an Aggregate and its Root could be a MemberSubscriptions aggregate. The information needed to make a successful payment would be supplied to a command on the Aggregate Root, `ApplyMemberSubscriptionPayment`. Inside the aggregate, there may be many steps of the process like gaining access to the credit card processing system, and more. The world outside only knows that it

can call `ApplyMemberSubscriptionPayment` and it will either return successful or as a failed attempt. It could have information about the condition of the failure that allows the calling code to determine if trying again is a business-friendly option. It may even have an event that posts to let the rest of the systems that care to know that the payment was accomplished so that notifications and more can be automated.

### **3.2.5 Entities Give Identity**

Entities are classes that have identities generally in the real world but could be a manufactured reality as well. The main idea here is that a John Smith in New York and John Smith in San Francisco are not the same people, and each have their own identities and need their own identifier. This allows us to work with the correct data for the correct John Smith. Entities help us remain safe as we remain adaptive since we know that an entity is not just some random data object. Entities deserve careful attention when we make modifications, especially in this day of regulation and information safety. Adaptive safety is a key to making sure we make quality changes, not just easy changes.

### **3.2.6 Value Objects Reduce Cost**

When we can, we should favor Value Objects over Entities. Value Objects provide us with a disposable nature to our software. An example is the object that holds a value for the hex color #E3242B (*Red Rose*). That object is the same as another object in memory for the hex color #E3242B. The more we rely on reusable value objects, the less we care about having the same number of those value objects as elements that need them, the more we can just use a suitable and available value object with the data we want.

There is no reason to have 50,000 objects with the same hex value for red when we need to consume it 50,000 times, unless there is a performance reason, we can use one. Value objects are easier to work with and will reduce complexity by the fact that we don't have unnecessary identifiers on these objects. Additionally, it is commonly understood that maintaining Value Objects costs less than maintaining Entity types given the disposable nature of a Value Object.

### **3.2.7 Domain Services**

The term service is used a lot in a lot of IT departments, and we do not have a common industry understanding of what that term means. In DDD terms, a Domain Service is a functional capability that does not require storage of transactional entity data but does perform a calculation or some other evaluation on input data. These are often standalone methods or functions. This building block should be used only when the method or function has no true entity, or value object home.

These services can be quite alluring to use, but it is important to realize that they in themselves can start the slippery slope of a big ball of mud. When we think we have a domain service, we should ask where the data is coming from that is being calculated or evaluated. If it is all from the same entity, maybe that function should be attached to that entity.

For example, instead of having a BillPay Domain Service as an aggregate all its own, maybe there should be a command "PayInvoice" on the aggregate entity "CustomerInvoice" with a command at the Aggregate Root that makes paying the invoice possible from the invoice. This process is likely not a Domain Service either because it tracks and saves some domain data.

An example of a domain service could be a calculation that is needed by many aggregates such as determining the roadway distance between two points on a map. That distance is needed to accomplish other things, but the distance calculation is utilized by a lot of other processes. It takes in two points, goes through the process of learning the shortest distance and then returns it to the caller. We never need to save that data when it is calculated.

### 3.2.8 Events Record History

Events are an amazing and growing concept in the custom software industry. Events, from a DDD or distributed systems perspective, are a record of an occurrence in the past. Once that event happened it is part of history. It happened and that record is not something that can change. Bank accounts are a good example. If a deposit was partial, then another deposit would produce a second event that would bring the balance to the correct value. We cannot change the first deposit, it happened. We can only react to it. Events are powerful in that they draw a line in the sand of reality. We know an event happened because it is emitted and stored. Events can trigger new commands for automation, or they can help a user know what happened.

Events help bring visibility to the computational flows of systems allowing us to know more about what we need to change when we are adapting to the future. EventSourcing strategies store events as they occur on an aggregate so that we can rewind and fast-forward through historical records of that aggregate to learn the steps that the data took to get to some state. Event Sourcing is a practice that the DDD community has been learning and growing with for many years. This concept of Event Sourcing is made easier by understanding DDD and Command Query Responsibility Separation (CQRS). CQRS at its heart is a pattern that separates the paths of the workflow that changes data from the paths that report it.

Domain Events are powerful, but the management of an event driven system is not trivial. Care and attention should be paid to the process of understanding how Events are triggered and how they are used to communicate to the rest of the system.

## 3.3 Eventstorming A Domain

The key to adaptive systems through DDD is that the domain is pulled into focus so that your software is so aligned that when the business changes, the software can change right along with it. To accomplish this, you will want to know what the Domain is doing and what it needs. Eventstorming and related paradigms have become popular in some circles to assist with this process.

If you have not heard about the orange sticky revolution started by Alberto Brandolini, let's introduce you to Eventstorming – a tool to make DDD even easier. Eventstorming by its very nature is best facilitated by someone with training. Workshops are best started with a group of Domain Experts helping everyone understand domain flows.

We can say a lot about EventStorming, however, Alberto Brandolini and Paul Rayner each wrote great books on the topic. Here, we will just talk about the high-level understanding of the tool. In general, EventStorming Facilitators will take teams through three levels of Eventstorming: Big Picture; Process Modeling; Software Design. Let's look at each at a high level.

## 3.4 Big Picture Eventstorming

During a Big Picture Eventstorming workshop, Domain Events are the focus. The Domain Experts write on the orange sticky notes all the Domain Events they know about for the process that we are designing. No order to start with is needed. Just get them all up on the board. Back to billing for example, some events might be, Bill Generated; Bill Sent; Bill Paid and so forth. Once the energy for Domain Events starts to subside, the team will be directed to put them on a timeline to get a sense of the flow.

## 3.5 Process Modeling Eventstorming

More types of process stickies are introduced. *Personas, Commands, Processes, Views, Business Policies and Questions* get added to the flow. These processes consider manual steps as well as automated steps. The facilitator shows a legend and soon the Domain Experts, if they are interested, will start to learn how to describe their processes in a way developers can more easily understand than typical requirements documents or user stories.

### 3.6 Software Design Eventstorming

Developers will take the process model and work to make a software design. This will coalesce the data and the behaviors where needed and decouple them where needed. The aggregates we discussed earlier will emanate from this process. The language can be OO or Functional. Java, Ruby, or .Net. A solution template that is DDD honoring will make applying these designs to code much easier. This is how we can save time and money the way UI developers do with UX designs.

## 4 Applying DDD To the SDLC

Adding a design step to a process that never made room for behavioral design could prove challenging. Let's explore that idea.

### 4.1 Project Management Education

Project Managers have struggled with managing software projects, especially if they were trained to manage projects that fit the manufacturing world. Managing software projects with a design step can help since design is also a manufacturing concept. However, it will still be important to learn how to manage *thought work* efforts for the unique challenges it brings as compared to manufacturing project management. If the Project Manager learns what DDD is, this can help them move toward a common language with the designer/developers putting these principles to work.

### 4.2 Some Executives Need to Champion Quality Behavioral Design

Many developers need permission to do good work with new steps. They can tend to feel like quality is not important more than quantity in many corporate cultures so they will not gravitate to new concepts like DDD that provide quality and eventually quantity. Executives need to find a way to message that they prefer a good quality adaptive product development mentality, because it provides for more capability in the future for less.

### 4.3 SDLC Upgrades

The main SDLC upgrade is the ability to accomplish a good design. When there are questions, the developer needs to have permission to ask for a clarification of a design. The Domain Expert needs to be part of the team, and the developer needs to be able to ask questions. Architectural decisions will come from these conversations so it will be important to make sure impacted teams are aware of those.

### 4.4 DDD Team Formation Considerations

Teams come in all shapes and sizes. It is generally best to keep team sizes small: 3-8 people. DDD will be more adaptive when the ones who create aggregates or at least maintain them have a bit of deep knowledge of the domain. They will make better decisions with the design because they understand that aspect of the domain. If they only have only a few team members, then collaboration is easier. If 95% of their changes are inside the aggregates they work on, their daily flow will be faster.

## 5 Conclusion

The way many craft software today with an emergent design model often can create stress on the SDLC. If we intertwine Tactical Domain-Driven Design, we can reduce the amount of churn needed to create the solutions we need to be successful. We can encapsulate behaviors into Aggregates that make change more predictable. Taking time to plan a solution architecture by aligning it to the domain makes change a little more manageable and can reduce waste. Eventstorming can make that process more achievable in an enterprise setting. Designs can flow into software templates that are structured for DDD solutions.

## References

- Abraham, Max. 2022. "Waterfall Methodology – Ultimate Guide." *Management.org* (blog). March 21, 2022. <https://management.org/waterfall-methodology>.
- "Agile and the Long Crisis of Software." n.d. Logic(s) Magazine. Accessed August 19, 2024. <https://logicmag.io/clouds/agile-and-the-long-crisis-of-software/>.
- Brandolini, Alberto. 2015. "Introducing EventStorming." Leanpub. September 9, 2015. [https://leanpub.com/introducing\\_eventstorming](https://leanpub.com/introducing_eventstorming).
- Evans, Eric. 2003. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston, MA: Addison Wesley.
- Forsgren, Nicole, Jez Humble, and Gene Kim. 2018. Accelerate: The Science of Lean Software and Devops: Building and Scaling High Performing Technology Organizations. It Revolution Press.
- Jones, Capers, Olivier Bonsignour, and Jitendra Subramanyam. 2011. *The Economics of Software Quality*. Boston, MA: Addison-Wesley Educational.
- "Manifesto for Agile Software Development." 2001. Agilemanifesto.org. Accessed August 14, 2024. <https://agilemanifesto.org>.
- "Poor Software Quality Can Cost Time and Money, Straightforward Solutions Are Available." 2023. CSO Online. March 13, 2023. <https://www.csoonline.com/article/574723/poor-software-quality-can-cost-time-and-money-straightforward-solutions-are-available.html>.
- Rayner, Paul. 2019. "The EventStorming Handbook." Leanpub. February 15, 2019. [https://leanpub.com/eventstorming\\_handbook](https://leanpub.com/eventstorming_handbook).

# Vulnerabilities in Deep Learning Language Models

Security Risks and Mitigation in Non-Federated, Federated and Decentralized Training

**John Cvetko**

John.Cvetko@TEKAsc.com

**Bhushan Gupta**

bhushangupta51@gmail.com

## Abstract

Deep Learning Language Models (DLLMs), particularly those based on Generative Pre-trained Transformers (GPT), have significantly advanced the field of natural language processing. As these models become more integrated into practical applications, they expose new security vulnerabilities and increase the attack surface for enterprises. Understanding and addressing these vulnerabilities is essential for ensuring the reliability and confidence in these systems.

This paper examines the vulnerabilities associated with both federated and non-federated training methods, including data poisoning, model update manipulation. Additionally, we discuss the need for robust detection and mitigation strategies to address these risks, ensuring the safe and secure deployment of AI systems in real-world environments.

## Biography

*John Cvetko*

*As a Principal at TEK, Mr. Cvetko works with companies and government agencies to improve their organizations by helping them manage the IT challenges they face. He focuses on applying state of the art solutions that support business goals and objectives. For the last 12 years he has primarily worked with state governments assessing and modernizing large enterprise software systems. He has worked with the state governments of Washington, Oregon, Colorado, North Carolina, North Dakota, and Utah. He has consulted for firms such as Gartner, Boeing, and MAXIMUS, and earlier in his career he has held program and systems engineering management positions at Tektronix, PGE/Enron and ASCOM.*

*Bhushan Gupta*

*An international speaker and a WebApp security researcher, Mr. Gupta is the principal consultant at Gupta Consulting, LLC. In WebApp security his research areas are infusing security in SDLC, OWASP Top10, Risk Analysis and Mitigation, Attack Surface Measurement, and Static and Dynamic Application Security Analysis. As one of the leaders of Open Web Application Security Project (OWASP) Portland Chapter, he is dedicated to driving web application security to higher levels and provides training workshops to corporations and non-profit organizations. He is also an invited speaker and a panelist in discussions for both application security and agile software development. Mr. Gupta serves as a board member of Pacific Northwest Software Quality Conference and was the Program Chair for PNSQC2022. He has also been a member of the Program team for the OWASP Global AppSec Conference 2020*

# 1 Introduction

To discuss the training vulnerabilities in a Deep Learning Language Model (DLLM), it's important to understand the text processing pipeline and training process. The pipeline includes three main components for data processing which are the Input, the Transformer, and the Output stages. These 3 three stages represent the "model" or the brain of the system. The model can be adjusted or tuned to a specific purpose by the approach and the data utilized in the training .

The training approach is also influenced by the purpose of the Large Language Model (LLM) and the needs of the users, for example a system designed to provide general use by the public will be much different than a system shared by multiple companies that are collaborating to create an industry specific model. To satisfy these needs designers may deploy federated and non-federated training configurations. Each of these configurations has its strengths and weaknesses relative to security and privacy. Depending on the training approach taken the systems may be more vulnerable to data manipulation while others are vulnerable to model manipulation.

This paper will first outline the basic workings of a model, training phases and then the training configurations before we highlight a few specific attack types based on their training data or corpus and model refinement.

## 1.1 Understanding the Processing Pipeline

This section outlines the primary stages of the processing pipeline at a high-level: Input Processing, the Transformer Stage, and Output Processing. These elements together make up a model that can accept user input and convert it to machine readable format. This information is then processed utilizing a neural network capable of keeping all the content in context. Once processed it is converted back to text that is output to the user.

**Input Processing:** The first step involves cleaning and preparing raw text data. This includes removing irrelevant characters, normalizing, and converting the text into a numerical format through a process called tokenization. The tokenized data is then further prepared for the Transformer by padding the tokens and masking irrelevant parts of the data. These preprocessing steps are crucial for ensuring that the data is in a format that allows the transformer to interpret then predict the proper response.

**Transformer Stage:** At the core of models like ChatGPT is the Transformer stage. The transformer concept was a breakthrough in the evolution of LLMs. The best way to quickly describe the transformer stage is through an analogy.

Imagine reading a novel. Each word, sentence, and paragraph provides more details about the plot. As you read and better understand the context you instinctively focus on key phrases and details, connecting them to earlier parts of the story to understand the plot as it develops. At the end of each chapter, you pause to reflect on the plot, piecing together the information and anticipating what lies ahead.

Traditional language models, like Recurrent Neural Networks (RNNs), read and present text word by word, similar to word prompts being suggested when you are writing a text message. This sequential processing can remember and connect information from earlier parts of the text message.

Transformers, on the other hand, take a more holistic approach. The transformer sees an entire sentence and keeps it in context to the paragraph, chapter, and book. It uses an "attention mechanism" to weigh the importance of each word in relation to all others, allowing it to capture long-range dependencies and understand the context of each word more fully. This helps the Transformer generate more coherent and contextually relevant text, making it a powerful tool for language tasks like translation, summarization, and even creative writing.

**Output Processing:** The final stage involves converting the model's numerical outputs back into human-readable text. This stage is critical for ensuring the generated output is presented in contextually accurate sentences.

By understanding these stages, we can better identify where and how DLLMs are vulnerable to attacks, setting the stage for the subsequent discussion on specific adversarial threats and mitigation strategies.

## 1.2 Training Phases and Configurations

Training a large language model (LLM) like ChatGPT involves several key phases, beginning with data preparation and followed by pre-training, fine-tuning, and finally Reinforcement Learning from Human Feedback (RLHF), see table below. Outlining these phases at a high level will help the reader to understand how the data curated and condition before using it to train the LLM. This data is specifically curated for each stage of the training. The refined data is necessary as the models' capabilities increase throughout the process. Currently the training of a generative LLM is both an art and science because of our current understanding of the technology. Achieving the right balance between providing the best data to a model in manner that ensures the best outcome at a reasonable price is challenging. When balanced properly, bias, hallucinations, security and privacy can be managed at a level acceptable to the designers.

**Data Preparation:** The first phase involves collecting and preparing the data for training. This includes gathering large datasets from sources like the internet, cleaning the data by removing irrelevant or erroneous information, and processing it into a format suitable for training. Tasks such as tokenization, normalization, and data splitting are performed during this phase. Proper data preparation is essential for training on high-quality data, directly impacting the model's performance.

Table: Training Phases for Medical Public Query

Training Phase	Data Sets	Result
Pre-Training	Broad Dataset (Wikipedia, books, etc.)	Basic Linguistics
Fine Tuning	Focused dataset (medical journals, etc.)	Fine-tuned knowledge
Reinforcement Learning	Human Feedback (Doctors, Nurses approvals)	Practical knowledge

**Pre-Training:** After data preparation, the model undergoes pre-training using self-supervised learning. During this phase, the model is trained on vast amounts of text data to learn grammar, facts, and reasoning by predicting the next word in a sentence. While pre-training is crucial for building the model's foundational knowledge, it is susceptible to attacks that can manipulate the learning process, such as model poisoning.

**Fine-Tuning:** Following pre-training, the model enters the fine-tuning stage, a form of supervised learning. Here, the model is trained on a specific, curated dataset with human-labeled examples, aligning it with desired behaviors and tasks. Fine-tuning allows the model to specialize in particular domains or tasks, enhancing its accuracy and relevance. However, this phase is also at risk of fine-tuning attacks, where an adversary could introduce biases or malicious behaviors into the model by manipulating the fine-tuning data.

**Reinforcement Learning from Human Feedback (RLHF):** The final phase involves refining the model's responses using human feedback. Humans rank the model's outputs, and these rankings are used to fine-tune the model further. RLHF is crucial for aligning the model's responses with human preferences, but it also introduces risks if the feedback process is compromised.

## 2 Training Configurations: Non-Federated, Federated, and Decentralized

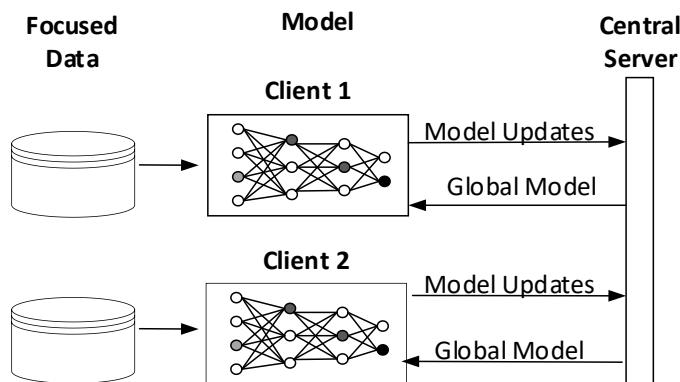
As mentioned earlier in this paper different training configurations are employed based on the intended purpose of the LLM. For example, a company may provide a generative LLM as a service to the general public like ChatGPT while other organizations may want to partner to create a robust common model for their specific industry. These two use cases have different requirements for privacy, and control that may be satisfied using a different training configuration. The most common configurations are non-federated, federated that is centralized and a decentralized or in peer-to-peer configuration.

**Non-Federated (Local) Training:** In this traditional approach, the data and model are centralized providing maximum control. While this method simplifies data management and processing, it also makes the system more vulnerable to traditional data breaches and model inversion or the reverse engineering of the training data through output of queries.

**Federated (Centralized) Learning:** Federated learning offers a more privacy-preserving alternative, see diagram below. In this configuration, data remains on the local devices (clients), and only model updates, such as gradients are sent to the central server. In machine learning, parameters are the internal settings of a model, such as weights and biases, that are fine-tuned during training, while gradients are the calculated values that guide how these parameters should be adjusted to minimize the error and improve the model's performance.

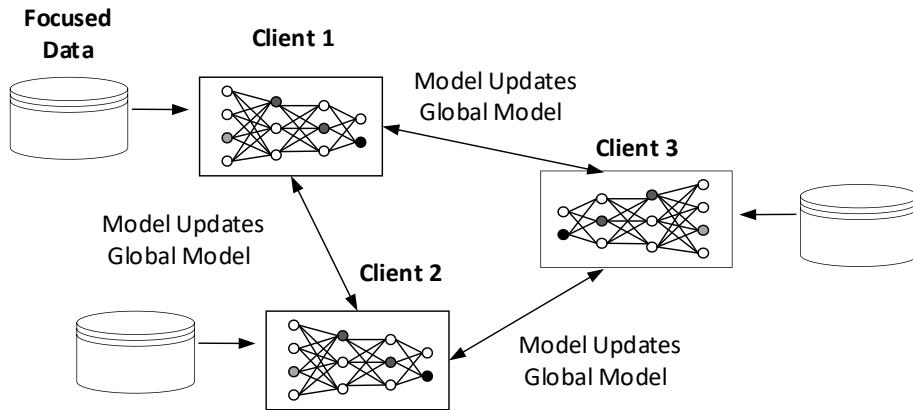
Updating the central server with only updates reduces the risk of data breaches by keeping the data decentralized and theoretically more protected. However, it introduces new challenges, such as model update poisoning, where malicious changes are made during model updates; Byzantine attacks, where some participants in the system act maliciously; and gradient leakage, which occurs when sensitive information is unintentionally revealed during the process of combining data from different sources. As Kairouz et al. (2019) highlight, federated learning introduces a delicate balance between privacy and model robustness, necessitating ongoing research to address these challenges effectively.

Diagram: Federated Configuration



**Decentralized Learning:** A more advanced approach is decentralized or peer-to-peer learning, see diagram below. In this setup, there is no central server; instead, clients (or nodes) communicate directly with each other to share model updates and collaboratively build a global model through consensus. This fully decentralized configuration enhances privacy and reduces the risks associated with a single point of failure. However, it also presents unique challenges in maintaining the integrity of the training process. As discussed by Kairouz et al. (2019), decentralized federated learning is particularly susceptible to Byzantine attacks, where malicious clients may attempt to disrupt the learning process by sending false or misleading updates. The absence of a central authority complicates the detection and mitigation of such attacks, making robust strategies essential for the secure deployment of AI systems.

Diagram: Decentralized Configuration



The table below summarizes the strength and weakness of each model for simplicity.

Table: Training Configuration Strengths and Weaknesses

System Type	Strengths	Weaknesses
Non-Federated (Centralized)	High efficiency and performance due to centralized computation and data access. Easier to implement and manage compared to distributed systems. Can leverage powerful centralized resources for complex tasks.	Single point of failure: Vulnerable to system outages or attacks on the central server. Privacy concerns: Data is collected and stored in a central location, increasing the risk of data breaches and privacy violations. Data ownership and control: Users have limited control over their data, which is stored and managed by the central entity.
Federated	Enhanced privacy: Data remains on local devices, reducing the risk of data breaches and privacy violations. Improved efficiency for distributed data: Leverages local computational resources, reducing communication overhead. Scalability: Can handle large numbers of devices and data sources effectively.	Communication overhead: Requires frequent communication between the central server and local devices, which can be a bottleneck. Heterogeneity challenges: Different devices might have varying computational capabilities and data distributions, affecting model training and performance. Potential for bias: Local models can be biased towards their local data, affecting the overall model's fairness and generalizability.
Decentralized	No single point of failure: Increased resilience to system outages and attacks due to distributed architecture. Enhanced privacy and data ownership: Data remains entirely under the control of individual users or nodes. Increased trust and transparency: Eliminates the need for a central authority, promoting trust and transparency among participants.	Communication and coordination challenges: Requires efficient communication and coordination mechanisms between nodes, which can be complex to implement. Slower convergence: Reaching consensus and training models might be slower compared to centralized systems. Security risks: Vulnerable to attacks targeting individual nodes or communication channels.

## 2.1 Targeted Training Attacks

The training phase lays the groundwork for an LLM's capabilities. Any weaknesses introduced during this stage can have significant consequences, impacting the model's performance, reliability, and potential for misuse.

The training process for an LLM is complex and requires massive datasets utilized as source information for multiple training phases each having multiple iterations for tuning. This creates a long complex chain of events that allows many points for malicious attacks.

Additionally, the growing popularity of federated and distributed learning with many different entities each having security limitations in their enterprises introduces further vulnerabilities. The training phase vulnerabilities can be segmented into two broad categories: data and model poisoning.

**Data Poisoning:** This type of attack involves the deliberate introduction of malicious or corrupted data into the training set. The goal is to manipulate the model's learning process, leading to biased, inaccurate,

or even harmful predictions. Data poisoning is especially prevalent in scenarios where training data is sourced from untrusted or crowdsourced locations, making it easier for adversaries to introduce compromised data without detection.

**Model Poisoning:** In contrast to data poisoning, model poisoning involves tampering with the model itself during the training process. Attackers may manipulate the model's parameters, gradients, or updates, particularly in federated or decentralized learning environments. This can lead to the model developing vulnerabilities, biases, or even backdoors that can be exploited later. Model poisoning is insidious because it directly alters the model's behavior, potentially going unnoticed until the model is deployed.

As Zhang et al. (2020) highlight in their survey on adversarial attacks in natural language processing, these types of attacks are not only pervasive but also varied in their approach and impact. Understanding the range of adversarial strategies is crucial for developing defenses that can protect against these threats.

Together, these attacks pose severe risks to the integrity and trustworthiness of machine learning models. This is particularly concerning in applications where accuracy and reliability are crucial, such as in healthcare, finance, or autonomous systems.

## 2.2 Data Poisoning

Data poisoning is a type of adversarial attack where an adversary intentionally introduces corrupted or malicious data into the training dataset with the objective of manipulating the model's learning process. This can lead to the model making incorrect predictions, introducing biases, or becoming more vulnerable to subsequent attacks. As illustrated in the table below, various forms of data poisoning attacks differ in their ease of execution, detection, and impact. For this paper we'll utilize Stealth Poisoning to characterize a data manipulation attack.

Table: Characteristics of Targeted Data Attacks

Attack Name	Ease of Execution (Internal)	Ease of Execution (External)	Detection Difficulty	Impact	Most Susceptible Configuration
Label Flipping	Easiest	Easiest	Low	Low to Medium	Centralized
Stealth Poisoning	Easy	Moderate	Very High	Medium	All
Targeted Poisoning	Moderate	Hard	Low to Moderate	High	All
Optimization-Based Poisoning	Hard	Hardest	Low to Moderate	High	All

## 2.3 Stealth Poisoning:

Stealth poisoning involves subtly introducing malicious data into the training set, allowing attackers to influence the learning process. This type of attack is designed to add insignificant data to a dataset that blends in seamlessly with the legitimate training data. Cleaning and data validation are important; however, some amount of noise is desirable to make the model more robust. When deployed in production the model will inevitably encounter data and learning to manage the noise is a desirable feature. Datasets that are not sufficiently noisy may allow the model to memorize the dataset causing overfitting. Overfitting can lead to inaccurate predictions when the model isn't sufficiently trained to manage the noisy nature of real-world conditions.

The goal of the stealth attack is to cause the model to make incorrect predictions, introduce biases, or become more vulnerable to subsequent exploitation. This type of attack is especially common in scenarios where training data is sourced from untrusted locations.

**Example:**

An internal or external attacker has access to the training data for a credit scoring model. The attacker introduces data that falsely indicate high credit scores for individuals who would typically be considered high-risk. As a result, the model could be manipulated to approve loans for high-risk individuals, leading to financial losses.

## 2.4 Detection and Mitigation

Detecting stealth poisoning requires rigorous data validation and anomaly detection techniques. Stealth poisoning attacks may evade basic checks and depending on the confidence in the source data, advanced anomaly detection may be warranted. Even the most sophisticated techniques are not without their challenges for example, patterns in the data may be incorrectly flagged as malicious causing false positives. Advanced techniques are also costly, time-consuming and resource intensive. Organization with a high-risk tolerance may accept the risk with the expectation that the fixes can be applied later if and when inaccuracies are identified in production.

Inaccurate predictions and unexpected behavior of the model in production may be an indication that the dataset was intentionally corrupted. To mitigate this issue outlier-resistant algorithms can be deployed to improve the model's ability to manage noise. This approach to refining the model in this manner is also costly and erodes the confidence of the users.

Another technique to improve the robustness of the model is to train it using adversarial scenarios. This approach intentionally exposes the model to subtle changes in the data. By training the model to detect unusual data it can flag the data for further analysis. Again, these techniques are costly and depending on the organization's budget or risk tolerance they may be excluded.

Together, these detection and mitigation strategies form a more sophisticated and layered approach to defending against data poisoning attack. The treatments help to preserve the reliability and security of the model during the training phase however they are just a few of the many tools needed to guard against attackers. Inevitable attackers will improve or develop new methods and more cost and time will be needed to manage the new threats to source data.

## 2.5 Most Common Model-Targeted Attacks

Model Update Poisoning is a type of adversarial attack that also targets the model training process. These attacks are more prevalent in distributed or federated learning environments because the model is intended to be common. Creating a common model requires multiple parties to contribute to the refinement of the model. The parties will need to move the model updates from one entity to another as it's being updated. These attacks can be targeted towards the global model in the central server or the models at the client nodes. The adversary looks to manipulate specific components of the model such as gradients or model parameters.

The attacker's goal is to corrupt the shared or global model by introducing malicious changes during the update process. The table below, outlines a number of model-targeted attacks that vary in their ease of execution, detection, and impact across different configurations. For instance, fine-tuning attacks are the easiest to execute and have a medium to high impact across all configurations, while Byzantine attacks, which are harder to detect, pose significant risks in federated and decentralized systems.

Table: Characteristics of Model-Targeted Attacks

Attack Name	Ease of Execution (Internal)	Ease of Execution (External)	Detection Difficulty	Impact	Most Susceptible Configuration
Fine-Tuning Attacks	Easiest	Moderate	Medium	Medium to High	All
Gradient Manipulation	Easy	Moderate	High	High	Federated
Backdoor Attacks	Moderate	Hard	Low to Moderate	High	All
Byzantine Attacks	Moderate	Hard	Very High	High	Federated/Decentralized

### **Fine-Tuning Poisoning:**

Fine-tuning attacks occur during the fine-tuning phase of the training process; this attack adjusts and refine the model using the smaller client dataset. In this phase, an attacker can introduce malicious behaviors, biases, or vulnerabilities into the model by manipulating the data.

For example, an attacker might subtly alter the fine-tuning dataset to alter the model sufficiently to cause the model to produce skewed or harmful outputs in specific scenarios. Alternatively, the attacker could embed specific triggers during fine-tuning that cause the model to behave maliciously when those triggers are encountered in real-world use.

Fine-tuning attacks are particularly dangerous because they can be relatively easy to execute, especially in environments where the fine-tuning data is not closely monitored. Additionally, these attacks can be difficult to detect, as the changes introduced during fine-tuning may be triggered at a later date or might only manifest under specific conditions, making them appear as normal variations in model behavior.

## **2.6 Byzantine Attack**

In the context of federated or distributed learning, Byzantine attacks occur when one or more clients in the system are compromised and behave maliciously. Clients in distributed environments train their local models on private data. These clients periodically send model updates again, typically gradients or parameter changes, to the central server or in peer-to-peer configurations directly to other clients for aggregation. In a Byzantine attack, the compromised client(s) sends updates that mislead or contaminate the common model which supports many organizations.

These attacks can be complex due to the variability built into the attack. For instance, the compromised clients might only send corrupted updates occasionally to avoid detection, they may also coordinate with other malicious clients to amplify the attacks impact when triggered. These attacks are more difficult to execute however when they are deployed, they are very difficult to detect and mitigate against.

### **Example:**

Consider a federated learning system where three banks (A,B and C) are collaborating to train a model that detects fraudulent transactions. If an attacker compromises Bank A , they could send corrupted updates that incorrectly classify fraudulent transactions as legitimate to the other Banks as well. This could result in a global model that is triggered to not detect fraudulent activities for a specific group.

### 3 Detection and Mitigation

To combat model update poisoning, a combination of detection and mitigation strategies is critical. Anomaly detection can identify model updates that deviate from expected patterns using statistical or machine learning-based techniques. However, detection devices may not be tuned precisely or be sophisticated enough and may cause more challenges such as false positives, i.e., flagging legitimate updates as malicious. High rates of false positives can waste resources and reduce the confidence in the model's responses.

#### Breach case study

Finding real world case studies is extremely difficult because there is no mandatory reporting requirements. Additionally, the rapid introduction of LLMs is outpacing the ability for the industry to develop standardize safeguards or processes that can aid in mitigation.

One well known incident that could be classified as both data and model poison is the Microsoft Tay debacle. Tay was a chatbot deployed by Microsoft on Twitter platform with the expectation that it would train in an unsupervised manner in production.

When the public identified that it could be manipulated to generate inappropriate content more users joined in which resulted in an overwhelming flood of bad training data. Since the model was continuously learning in an unsupervised mode the poisoned data created a feedback loop. The model adapts to the poisoned data, which in turn influences the generation of further outputs that are consistent with the poisoned data, reinforcing the biased learning. In short, the model was intentionally manipulated causing a dramatic shift in the model's parameters.

#### Model Supply Chain

While not technically a model poisoning exploit the Hugging Face incident is a good example of vulnerabilities in AI supply chains. Hugging Face serves as a platform where researchers and developers can share and collaborate on pre-trained models and datasets. Over time a number of compromised models were uploaded to the site leading to unintended consequences for unsuspecting users who downloaded and integrated them into their projects. If a widely used model is compromised, it can have a cascading effect, impacting many downstream projects and applications.

### 4 Conclusion

This paper has explored the vulnerabilities in Deep Learning Language Models (DLLMs), with a focus on non-federated, federated, and decentralized learning systems. While these models offer significant advancements in natural language processing, they also pose substantial security challenges.

Non-federated learning systems are particularly susceptible to data poisoning of fine-tuning datasets. These attacks can severely degrade model performance, introduce biases, and compromise privacy.

Federated learning systems, which decentralize data storage by keeping it on local devices, enhance privacy but are not without their own vulnerabilities. They require a delicate balance between privacy and model robustness, this tension makes them vulnerable to model update poisoning and Byzantine attacks.

Decentralized configurations, while offering further privacy benefits, bring technology and collaboration complexity which further increases the attack surface during the training process.

In conclusion, while DLLMs hold great potential, the technology is in the rapid adoption phase of its lifecycle. This phase can outpace an organization's ability to manage the technology securely. It is essential to recognize and address the vulnerabilities relative to their training configuration and intended use.

## 5 References

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17) (pp. 6000-6010). Curran Associates Inc.
- Zhang, Z., et al. "Adversarial Attacks on Deep Learning Models in Natural Language Processing: A Survey." IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 7, 2020, pp. 2450-2467. Semantic Scholar, <https://www.semanticscholar.org/paper/Adversarial-Attacks-on-Deep-Learning-Models-in-A-Zhang-Sheng/652107ea8161f607e3bdabc89199e9ff2fdfd015>.
- Wallace, E., Feng, S., Kandpal, N., Singh, S., & Gardner, M. (2019). Universal Adversarial Triggers for Attacking and Analyzing NLP. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019) (pp. 2153-2162). Association for Computational Linguistics.
- Aidan, S., & Butman, M. (2021). Practical Byzantine Fault Tolerance in Federated Learning. In Proceedings of the 40th IEEE International Conference on Distributed Computing Systems (ICDCS 2021) (pp. 924-933).
- Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... & Yang, H. (2019). Advances and Open Problems in Federated Learning. Foundations and Trends® in Machine Learning, 14(1-2), 1-210.
- OpenAI, 2024. ChatGPT (Version 4) OpenAI URL: <https://www.openai.com/>

# Mutating Test Scenarios

**Explore the unknown by disregarding order and welcoming events**

**Baubak Gandomi**

Baubak@Gandomi.com

*"It bothered him that the dog at three fourteen (seen from the side) should have the same name as the dog at three fifteen (seen from the front)"*

JL Borges (*Funes the Memorious*)

## Abstract

Most testing, whether automatic or manual, strives to raise predictability. This is usually achieved by:

- Following a predefined set of steps
- Removing ambiguity in the findings by eliminating static

In this paper we make the point that these assumptions originate from manual testing and that blindly following such assumptions in automated testing will not provide any additional advantages apart from increasing the processing capacity.

Scenarios that follow a predefined set of steps are too simplistic, and they should be leveraged to go beyond their original intent. In other words, the purpose of test automation should be to perform tasks that cannot be done by manual testing.

We therefore propose a new testing method, which we define as "mutational" testing. In this method, the given scenario can, when needed, adapt, change its structure and order, i.e. "mutate", to address the challenges that are imposed on it. We also show that by injecting selected "noise" in the execution of a scenario, we open new opportunities in test domains.

We will focus here on two major domains, each an aspect of mutational testing:

- Events: Events taking place during the execution of tests
- Permutations: The user may take a different path than originally intended

The mutational test methods help solve problems such as:

- Software Migration testing
- Software Upgrade testing
- Chaos testing
- End-User testing

## Biography

The author, Baubak Gandomi, studied at the Mid-Sweden University, Universität Leipzig (Germany) and finally graduated from the university of Stockholm, Sweden with a degree in Computer Sciences.

He is currently a test automation architect at Adobe, a position he has held for the last 10 years. Apart from upgrade tests he is very interested in cross-product/teams testing and finding measurement methods for assessing functional and transactional coverage.

Baubak is currently residing in France with his family. In his free time, he practices and teaches the martial art of Ninjutsu.

## 1 Introduction

A lot of this work originates from addressing the complexity of testing systems and product upgrades. After each major upgrade, testers are required to check that certain workflows are still functioning correctly.

Product upgrades can be in many shapes and forms. At Adobe Campaign the notion of “upgrades” involves testing the upgrade of our product from one version to a newer one. Due to the complexity of our product, it could involve API, database and simple functional changes. Our goal was to ensure that processes and workflows starting before the upgrade could be successfully fulfilled after the upgrade.

Like many good ideas, its implementation, and subsequent presentations led to additional questions and problems.

This paper presents our attempts at creating a problem statement that properly represents these problems. It also shows how we have tried to leverage the concepts introduced by the Phased Tests for addressing these problems.

### 1.1 The Original Problem Statement

When dealing with our original problem, testing upgrades, we usually had two major problems:

- How does one make sure that the problem is related to the change, and is not inherent to the product?
- How can a problem be clearly identified and reproduced?

This led us to identify two underlying issues with the testing we were doing:

1. The Data problem
2. The Time problem

#### 1.1.1 The Data Problem

Traditional testing is insufficient because it often does not consider the data prior to the system change. Systems performing such tests usually manage a database that is regularly updated. Managing such databases is quite complex and is hard to scale. We identify the following problems:

- Mapping data to a specific test scenario is complex
- Mapping data to a specific product/application version is complex
- Updating the data is cumbersome

#### 1.1.2 The Time Problem

Most test scenarios imagine a static system. They will perform a set of actions and verify that the end result is what was expected. There are however, two problems with the traditional approach:

**Scenarios are not atomic in real life:** User Transactions are not usually atomic. The way we use services is rarely a sequence of events with a predefined time interval between each action.



Users rarely perform a series of actions in one go, and they interrupt the process for various reasons

**Other scenarios are usually being executed in parallel:** You are not the only person executing a scenario / transaction. It is quite likely that others will be performing the same workflow, and may be at different stages.



## 1.2 Phased Testing: A Solution to the Upgrade Problem

We developed a notion called Phased Testing to address these problems. In that solution, we first put forward a notion of a scenario as a template, which is central to this paper.

Phased Testing was based on two major concepts:

- A test scenario should be interruptible at any point
- A test scenario is declared only once, but we should be able to execute it with all the possible interruptions it may be subject to

When executed in the case of an upgrade test, the system will detect all the possible interruptions a scenario can have. It will then re-execute the scenario as many times as there are possible interruptions, stopping the scenario at the expected interruption point. It will then await the system change, only to carry on the scenarios where they left off.

This mechanic is called “Shuffling”.

## 2 Lessons Learned from Phased Testing – Enter Mutations

During the various presentations on the subject of Phased Testing and in the course of various development initiatives, we came to certain conclusions.

## 2.1 An Upgrade is just Another Event

Not all upgrades are so invasive as to interrupt the work on a product. In many cases this can be done with no downtime. In addition, if we want to look at other use cases, we need to look beyond upgrades as a phenomenon. Therefore, we came to the following conclusions.

- An Upgrade can be categorized as a System Change, which can itself be classified as an Event which causes an interruption
- Not all events are interruptive, nor are all they only “upgrades”. We should be able to measure their impact as well

## 2.2 Scenario Orders are Illusory

We always expect a scenario to follow a certain order of steps. In reality, there is no rule for a user or process to follow a specific order of steps. We tend to write scenarios to reproduce a sequence of events. However, a lot of bugs come from users taking unexpected paths.

## 2.3 Erasing Walls of Time Dimensions in Test Scenarios

We need to acknowledge that executing a scenario is just a snapshot of what one user will do at a given time to achieve a goal (aka use a functionality). However, we also need to acknowledge that there will be other users achieving the same goal, and who are simultaneously at a different step along a scenario. When an event occurs, the users may be affected differently by the change.

With “The Time Problem”, we acknowledge that an event can occur at any time along the scenario process, and that we will trace these paths so that we can correctly predict the effects of the events on the users.

## 2.4 Mutations as a Generalized Set of Conclusions

These ideas finally led to a new, generalized set of deductions:

- A scenario is a template that can, and should, be executed with different characteristics
- A scenario can be subject to different Events along its execution. Some are interruptive, some are not
- Scenario steps do not need to follow a predefined order

A scenario needs to adapt, change structure and order to correctly cater to requirements. In other words, we say a Test Scenario can “Mutate”. Even though a scenario can mutate, it still remains a set of steps achieving a goal. In the next chapter we will look more closely at what these mutations can be.

## 3 Approach

This paper will explore different types of Mutations, and how they can be implemented and used. As mentioned in the introduction, one of the goals of this paper is to leverage the concepts introduced by Phased Tests to enable Scenario Mutation. In this chapter, we will describe the mechanisms that permit us to implement the mutations to which a scenario can be subjected.

### 3.1 A Scenario and a Set of Steps

In order to allow tests to mutate in the way we want, our scenarios need to have steps that are clearly separated and have clear boundaries.

In many testing tools, such as codeless systems and Cucumber, there is a clear identification of steps, whereas in frameworks originating from unit testing, the notion of a defined step is not naturally implemented.

In the case of Phased Tests we opted for the Scenario as a Class and the steps were represented as Class Methods. Hereafter we will continue following that model.

### 3.2 Defining a Test Execution Mode

By default, a Test Scenario will follow the order of steps as defined by the developer.

We will provide the possibility for the user to change this behavior at execution time, so that a Scenario will mutate and diverge from the original scenario.

### 3.3 Test Shuffling and Shuffle-Groups

The concept of “shuffling” involves the multiple re-executions of a scenario, based on a stimulus or a requirement. In the case of Upgrades, the shuffling is based on the possible interruptions a scenario can be subject to whenever an upgrade happens.

Each re-execution or iteration is identified by what we call a Shuffle Group. The Shuffle Group also acts as a context in which the steps have a relationship and share context variables.

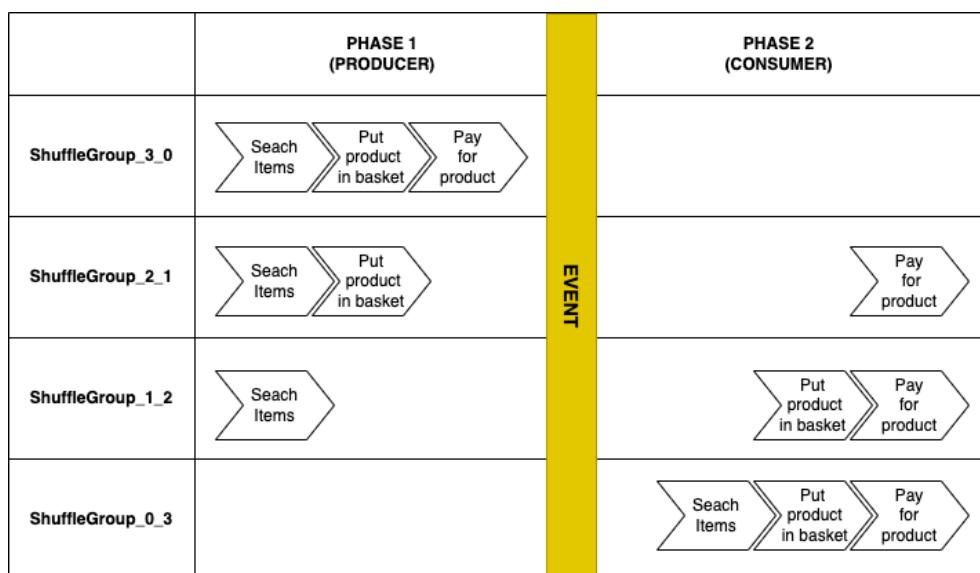


Figure 1 - Shuffle Groups and Phases

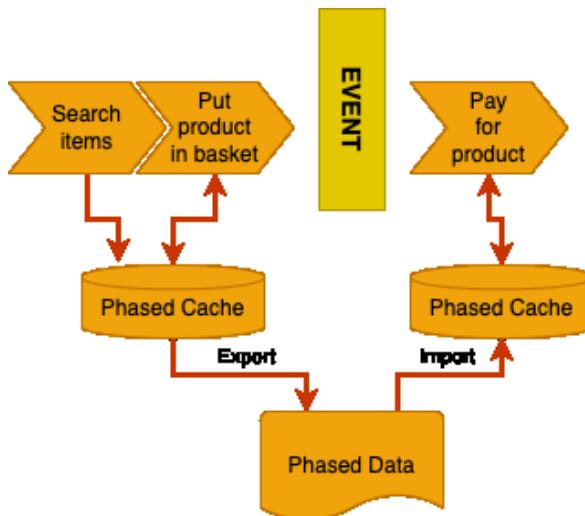
At each shuffle phase group, we stop the scenario at a different step. After the “Event”, for each Phase Group the scenario carries on where it left off. Figure 1 shows the shuffle groups in the case of an interruptive event. We divide the execution into two parts called phases, based on their execution relative to the event.

### 3.4 Producing Data and Consuming Data in Steps

In the context of Phased Tests, the scenario can be interrupted after a given step. This means that the test program is actually interrupted, and as a result the variables will no longer be in the memory. We need to store the values of the variables in long-term storage so that the scenario can carry on after it restarts.

When we “produce” data, it is stored in the context of its phase group. This means that any following step can access this data. Even if we have an interruption, the data is stored in a cache for later use.

We therefore introduced the notions of “produce” and “consume”. A step will produce data that is consumed by later steps.



*Figure 2 - Storing and maintaining the context between steps and interruptions.*

In Figure 2 above, we have represented how produced and consumed data is managed, even after we have interrupted the system to perform our change.

### 3.5 Test-Driven Mutations

When choosing to mutate, the scenario adapts to the mutation it is exploring. We call this a test-driven mutation as we manipulate the test scenario rather than the event or data. The events for example are simply triggered according to the test scenario steps.

## 4 Scenarios that Mutate

The notion of mutational testing is that the scenario is a simple executable scenario, which, when needed, can mutate and change its behavior.

As mentioned in the introduction, the mutational scenario can be used as a template to explore new aspects of your product. In this paper, we identify the following mutations:

- Data
- Events
- Permutations

### 4.1 Data-Driven Mutations

The most simple and common mutation is what is known as “data-driven” testing. In this mode the tests and scenarios are iterated based on a dataset. Typical examples of data mutations are:

- Data Providers
- Language and Encodings

Data-Driven Mutations will be covered summarily as it is a common practice in testing, and there is considerable documentation on this topic.

#### 4.1.1 Changing Test Data

This is not too uncommon as we sometimes need to execute the same scenario with different test data and inputs. This is either done by duplication of the scenario, or by using tools provided by the test framework.

Most test frameworks provide such capabilities, where a scenario will be re-executed for a given set of test data, iterating over each entry. Examples include:

- Parameterized tests in Junit
- Data providers in TestNG
- Data Tables and Scenario outlines in Cucumber

#### **4.1.2 Languages and Encodings**

It is often a requirement for many software functionalities to work in multiple languages, and to cater to different character encodings. This means that a test needs to be able to be executed multiple times based on the languages your functionality will be available in.

When you use random data for your tests, it is quite simple to change the encodings or the language of the test data. There are libraries such as Faker where test data can easily be switched between languages and encodings.

### **4.2 Event-Driven Mutations**

A consequence of our previous work on Phased Tests was that we generalized the approach by recognizing that what drives the test is an event.

Events can happen at any time along a scenario. This makes the reproduction and the full impact assessment of an event tricky. In order to create a reproducible and verifiable approach we can state that an event can happen Before, During, and After a scenario step.

With events we can identify two categories:

1. Interruptive Events
2. Non-Interruptive Events

#### **4.2.1 Interruptive Events**

Scenarios are rarely an atomic set of steps. Instead, people will perform actions according to whim and will abruptly interrupt the process. During the interruption, many events may happen; the system may be shut down or upgraded. We need to make sure that such major interruptions do not interfere with processes that have not yet come to an end.

Due to the nature of interruptive events, there is a N to 1 relationship between the scenarios and the event. We identify the following use cases when it comes to interruptive events:

- Database Migrations and Changes
- Cloud Migrations
- Major Software Upgrades or Migrations

The Phased Testing framework was originally devised for Interruptive Events, i.e. you need to stop a system so that you can perform some system change such as an upgrade. Once the upgrade is done, we expect that the users can carry on with what they were doing.

The execution of steps in interruptive events is divided into two phases depending on their execution relative to the interruptive event. The phase before the event is called “producer”, because the steps executed before the event produce data used after the event has taken place. Similarly, the phase after the event is called “consumer” because the steps rely on data created in the phase before the execution of the event.

#### 4.2.2 Non-Interruptive Events

In many modern systems, upgrades and system changes are performed with no downtime. This means that events will happen at any given time.

Another major domain that is covered by non-interruptive events testing is testing “resilience”. This means that they need to withstand events. A good example of this is a driverless car challenging a intersection. The goal of the scenario is for the car to successfully pass the crossroad. The events could be anything from a reckless pedestrian or car. We will have to adapt to the event and then finish the task of driving through the intersection.

As in the case of interruptive events, we never know when the event can happen, so we need to be able to simulate the event happening at all stages of a scenario.

The challenge here is reproducing a detected problem, and to identify the combination that causes a problem.

One may correctly suggest that we can simply add an event within a scenario. The problem with this assumption is completeness, as this will work for one specific step, it will not be efficient if we want to test the effects of an event along the whole process of the scenario. As in the case of interruptive tests, “shuffling” solves this problem. Shuffling will allow us to see the effects of an event at every step of a scenario.

The notion of non-interruptive events is quite close to chaos testing. While most Chaos Testing systems control the triggering of the events, we have opted to have a tight coupling between an event and the scenario. We will not examine if one solution is better than the other, as our focus is on seeing how far our approaches, as described in Chapter 3, can take us.

An event, as we have modeled it, will happen:

- Before a step starts
- During the execution of a step
- After the execution of a step

The challenge here is to be able to avoid slippage, or ambiguity between a step and the event. An event should not “slip” into a step it is not supposed to affect.

An event happening during a step needs to start after the step starts and has to be finished before the next step starts. An event happening between steps needs to finish before the following step starts.

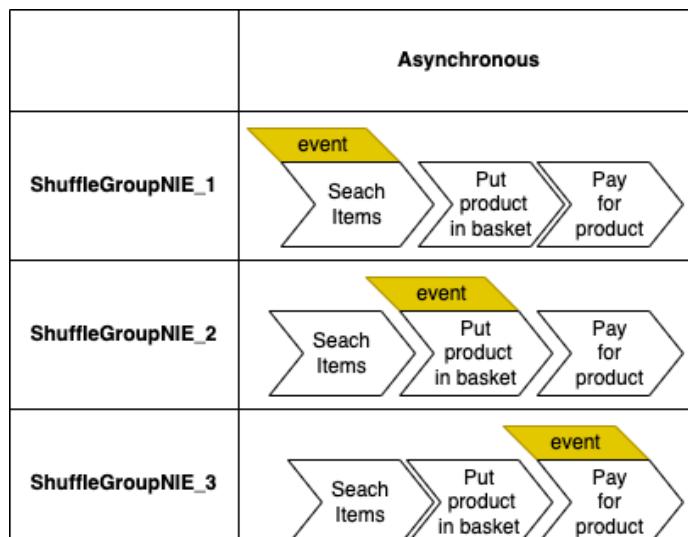


Figure 3 - Shuffling Tests during Non-Interruptive Events

This can be seen in Figure 3 we shuffle the scenario based on the step where we plan to inject the event. Each of these re-executions is identified by its “shuffle group”.

#### 4.2.2.1 Designing and Executing an Event

In our design it is the role of the framework to execute the event. This allows us to have better control of the orchestration and timing of the event in relation to the steps.

This requires events to follow a set of characteristics. We should have the following capabilities:

- Start the event
- See when an event has finished
- Access state information on the event

### 4.3 Permutations – Changing the Order

If we take the metaphor of a journey, there are many ways of looking at the process. The most obvious one is to reach a destination by the shortest path. Another way would be to take the more scenic path to get there. Both lead to the same destination.

As in the journey, we usually expect a series of steps leading to a destination. However, this is rarely predictable, and users typically take different paths in order to reach a specific goal.

If we consider that the purpose of the scenario is to reach an end-goal, the steps leading to the end-goal may assume a different meaning. It could be that the order of steps and the paths may become the test itself.

We believe that for a given scenario, we can identify a series of permutations, where some of the steps can be switched. In other words, the steps in a scenario need not be executed in the original order.

#### 4.3.1 Permutating a Scenario

The way we calculate permutations is based on two ideas: the definition order, and the dependencies between steps.

In Phased Tests, steps can communicate variables by storing them in their context. To store a variable, we use a method called “produce”, and the following steps can fetch the variable by performing “consume”. This creates an interdependency between steps, where steps that consume a scenario need to come after one that produces the variable.

Let us take an example as seen in Figure 4 below. We now add a new step to our web shop example: logging in.



*Figure 4 - A revised used case with authentication*

If we look at the produce and consume scheme, it will yield a process of the type illustrated in the figure below.

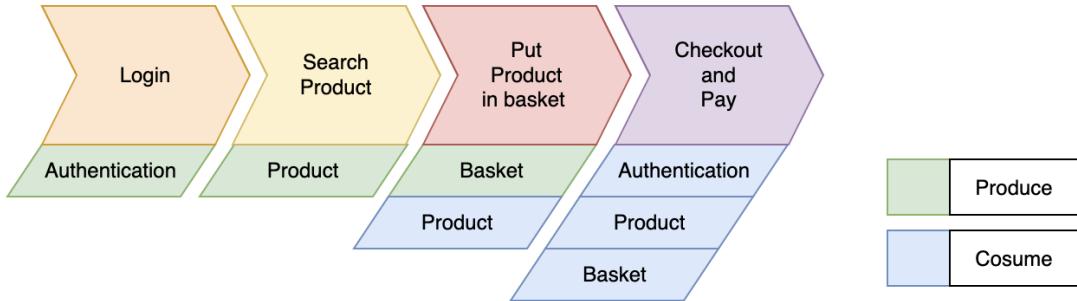


Figure 5 - The produce and consume schema for the web-shop scenario

In this model we make the following assumptions:

- We can search for a product on a web shop and add it to our basket at any time
- A payment requires a user to be authenticated
- Before paying, we make sure that we have selected the correct product

If we look at all the permutations, we can see that the steps “Login” and “Search Product” can be switched. However, Search Product needs to be before “Put Product in Basket”. We also see that “Login” can also happen at any step before the last one since performing the actual transaction requires authentication. If we permute this scenario we obtain the following cases as seen in Figure 6 below.

<b>ShuffleGroup_permutational_1_3</b>	Login → Search Product → Put Product in basket → Checkout and Pay
<b>ShuffleGroup_permutational_2_3</b>	Search Product → Login → Put Product in basket → Checkout and Pay
<b>ShuffleGroup_permutational_3_3</b>	Search Product → Put Product in basket → Login → Checkout and Pay

Figure 6 - The permutations of our scenario

#### 4.3.1.1 Limitations and Workarounds

A big problem with this approach is that the notion of “produce” and “consume” is not enough to give a clear view of the order of execution of the steps. Some methods will make changes to an object or even to the system, causing a state change which is a prerequisite in following steps. These changes may be required for the execution of next steps.

To resolve these kinds of issues, one must explicitly produce and consume these resources in the steps, so that the framework can correctly create the permutations. This can be done by introducing these state requirement checks in your steps.

### 4.3.2 Calculating the Permutations – A Deep Dive

As mentioned earlier, we look at the internal dependencies between the steps in order to calculate the possible permutations of that scenario. This is made possible because of the cross-step communication mechanism we described in Chapter 3.4.

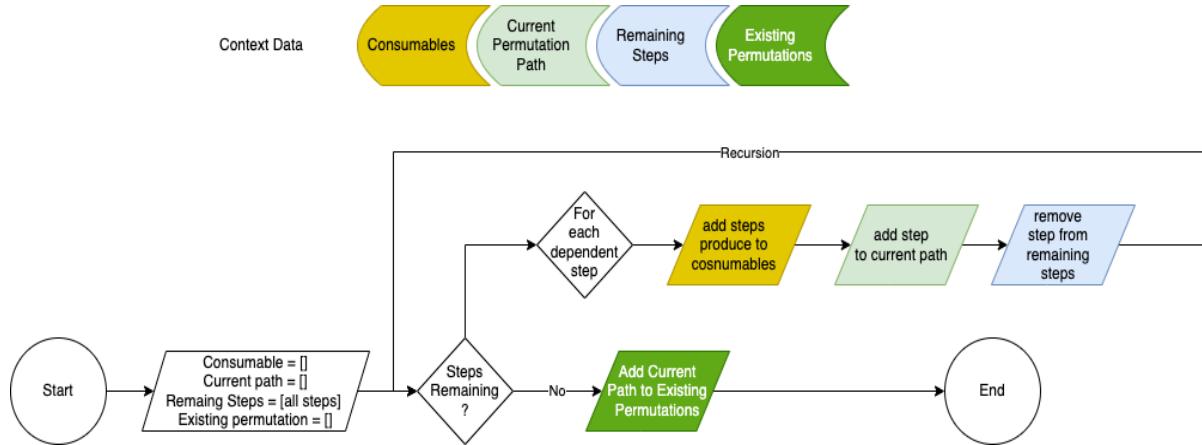


Figure 7 A flow chart describing the permutation calculations.

The algorithm performs the following steps

- Find the steps that can be executed with the given set of produced data.
- For each step start a recursion:
  - Add the data it produces to the set of consumables
  - Add the step to the current permutation path
  - Remove the step from the non-treated steps
- If there are no steps to treat, add the current permutation path to the stored permutations list

## 5 Conclusion

This work is the culmination of a simple project where we simply wanted to make sure we covered some use cases in upgrades. We believe that with mutational tests, automation tests will step beyond simple non-regression testing and will allow the finding of new paths for testing software. For example, with how events are addressed we provide users the ability to simply test hypothesis regarding the products stability and reactions to external stimuli. We hope this allows us to be more playful with the products we test, and to explore new ways to break the system we test.

The notion of Permutations is something we have been very interested in and have taken time to implement. In hindsight, we have also realized that the notion of permutations puts a greater responsibility on the test steps, as they need to function as independent and self-aware units. This means that they need to have enough information to let us know if they can be executed in a given context.

This problem is not something we usually deal with since scenarios have traditionally been atomic entities, whose steps were not originally intended to be separated. Despite this, we think that in the future such approaches will be more common. Many BDD frameworks are similar in their structure, and the step of writing a self-aware step is not too far.

As a follow-up we would like to see or implement these initiatives in BDD and Low-Code systems, where steps are naturally implemented.

Another potential for the notion of permutations and self-aware steps is if we break the boundaries of the scenario as a unit, and instead create new scenarios from the steps of different scenarios. This enables the discovery of new test paths that, as testers, we had not thought of, and which the users

are quite likely to use. Permutations beyond the steps of one scenario are technically possible and would make for a very interesting case study.

## Acknowledgements

The author of this paper would like to express gratitude to the following people (in alphabetical order):

- Charles McBride, for his kind and helpful remarks.
- Gary White, who yet again helped me with proof-reading and provided great insights.
- Nemanja Todorovic, for his great suggestions.

## References

Gandomi. 2022. "Phased Testing: Testing Systems Undergoing Change" PNSQC 2022 Proceedings: The Evolution of Quality: 40-56.

Cédric Beust, Hani Suleiman. 2007. "*Next Generation Java Testing: TestNG and Advanced Concepts.*" Addison-Wesley Professional

Marty Lewinter, Jeanine Meyer. 2015. *Elementary Number Theory with Programming*. Wiley

Baubak Gandomi. 2021. "Automated Upgrade Testing: A Process to Test and Validate Software Upgrades". Adobe Tech Blog, entry posted March 2021. <https://medium.com/adobetech/automated-upgrade-testing-a-process-to-test-and-validate-software-upgrades-349e647f34f4> (accessed March 11 2021).

The Phased Testing TestNG Implementation. <https://github.com/adobe/phased-testing>.

Netflix Technology Blog. 2011. "The Netflix Simian Army". Netflix Technology Blog, entry posted July 2011. <https://netflixtechblog.com/the-netflix-simian-army-16e57fbab116> (Accessed May 2023)

Paul Marshall. 2022. "Chaos Testing Stardog Cluster for Fun and Profit", StarDog Labs Blog, entry posted July 13<sup>th</sup> 2022. <https://www.stardog.com/labs/blog/chaos-testing-stardog-cluster-for-fun-and-profit/>

# Inside Scoop on Risk-Based Testing

**Robin F. Goldsmith, JD**

President, Go Pro Management, Inc.

[robin@gopromanagement.com](mailto:robin@gopromanagement.com)

## Abstract

Prioritizing tests based on risk is an essential aspect of all testing rather than a separate testing technique. Why then do so many people mistakenly think of it as something separate to be used only when your testing back is against the wall? Moreover, many find that conventional descriptions of how to assess and prioritize risk don't work in the real world.

This article reveals why typical risk approaches fall short and ways to make them actually work. It describes the conventional view of risk-based testing and why it's wrong, suggests how to take advantage of the fact that all testing is risk-based, and compares the value of various ways to assess risk for testing.

## Biography

*Robin F. Goldsmith, JD is one of the few with legitimate credentials in both QA/testing and requirements/business analysis/value, as well as project and process management/measurement/improvement. Uniquely positioned to help you get REAL value right results right, he works directly with project teams assisting, coaching, and training business, systems, and project professionals on risk-based Proactive Software Quality Assurance™ and Proactive Testing™, REAL requirements, project management and leadership, REAL ROI™, metrics, outsourcing, and process improvement. Go Pro Management, Inc. President and SQGNE past President and current Vice President, Robin is author of the Artech House book Discovering REAL Business Requirements for Software Project Success and the forthcoming Cut Creep—Write Right Agile User Stories and Acceptance Tests.*

## **Conventional View of Risk-Based Testing**

Testing is the main means of controlling software risk. Consequently, identifying and prioritizing risk is an important element of determining what to test and how much to test respective elements.

Seemingly related to this, the term “Risk-Based Testing” is widely-known and used. However, the term seems to be widely misused. I realized this thanks to repeated experiences with the following Dictated Project Constraints exercise in my *Managing Testing Projects* full-day seminar.

You are the Test Manager for a major system with a scheduled 6 months total project duration. Your Test Group is to do the System Test in months 5 and 6. You have the following concerns:

1. Two months of testing by your small group seems way too little.
2. You suspect the developers will be late getting the code to Test.
3. Previous projects had quality problems when testing was squeezed.

*Identify:*

1. *The points you would raise to convince management to give you what you need.*
2. *Management's likely responses.*
3. *Why?*

Participants typically cited the following points to convince management to give them what they need:

- Inadequate amount of testing in past led to trouble.
  - Need more testing.
- Two months testing is too little.
  - Need more test time.
- Test group is too small to do needed testing.
  - Need more testers.
- Many errors are in requirements and designs.
  - Involve testers earlier to make sure requirements are testable and known.
- Make Development deliver code in time to test it, preferably in multiple smaller and mainly earlier releases.
- Cut the functionality to what we can test in given time.

Participants uniformly agreed on management's likely response:

- **No!**
  - You're not getting more time or testers to enable doing more testing.
  - We're not changing development to involve testers earlier or to make Development deliver code earlier so it can be tested more thoroughly.
  - We can't cut functionality. It's all needed.
- Suck it up and do a better job of testing with the time and resources you've got.

Many testers in different classes and from different organizations all seem to have similar perceptions of testing's likely requests and management's likely responses to them. The testers tend not to have much insight into why management responds as it does except agreeing that management doesn't understand, or apparently value, testing. They also agree management should understand and value testing, and by implication should grant testing's requests.

At about this point in the exercise's discussion, since they're getting no satisfaction from management, *as a last resort* participants say something like,

- "Then, we'll do risk-based testing, just test the highest risks, not the rest."

### **Reiterating the Conventional View of Risk-Based Testing**

This common response to the seminar exercise demonstrates the common view of risk-based testing, that it:

- It is a special technique which somehow is different from other testing
- Occurs only in conjunction with explicit risk analysis
- Generally is used infrequently and as a last resort when other approaches have been exhausted.

I contend that this definition not only is wrong but indicates additional issues that limit testing's effectiveness. That is, risk-based testing is not special or different from other testing, need not be done explicitly, and should not be considered a last resort.

### **All Testing Is Risk-Based**

Rather than being special or different, basing testing on risk is inherent in all testing. That's because testing is our main means of reducing risk of systems and software. There's a simple relationship: the more risk, the more testing is needed.

Thus, whether or not recognized, conscious, or explicit: **all testing is risk-based!**

When we decide what to test, what to test first/next, how much to test it, and what and how much to retest, we're deciding based on the risks we perceive. The question is how adequately we've identified and analyzed the risks.

Too often, the answer is that we haven't identified and analyzed the risks very well. That shortcoming is exacerbated when the act of identifying and analyzing risks is not recognized, conscious, or explicit. Then it's much more likely that we overlook and misunderstand risks that testing could help reduce.

Testing's value is primarily driven by risk-based determination of what and how much to test. Only then is it worthwhile to address how best to perform relevant tests within budget, time, skill, and resource constraints.

### **Classical Risk Management**

All risk management involves six steps:

1. Identify potential risks
2. Prioritize the risks, qualitative & quantitative
3. Assess the causes
4. Accept risk or define a mitigation strategy
  - a. Avoid doing the risky thing
  - b. Control the risk's likelihood and/or impact
  - c. Transfer some or all of the risk
5. Define a contingency plan
6. **Implement** the strategy, monitor and control

How well potential risks are identified in Step 1 determines how effectively risks can be managed. Consequently, most risk management training emphasizes ways to identify potential risks; and most risk management weaknesses relate to missing or misunderstanding potential risks.

Testing is concerned with mitigating a risk's impact or likelihood by demonstrating whether the risk comes true in certain circumstances which affect the risk's impact or likelihood. When testing reveals a risk will come true, then further actions are needed to prevent the risk from occurring in production. Such actions include recoding, often in response to redesign.

Steps 1-5 are planning activities. Planning is necessary but not sufficient to manage risk. Thus, after planning, the next most important determinant of risk management effectiveness is how well the planning is implemented. Implementation includes monitoring and controlling performance of the planned risk management activities. In turn, that includes recognizing the need for and making relevant adjustments.

It's common to keep risk-related information in a "Risk Register." Whether or not explicitly designated as a risk register, it's valuable to capture the set of identified risks and analysis of them. Keeping this information together in identifiable form aids answering questions that may arise regarding the risks; and it provides a convenient way to guide and track risk-related activities, especially testing.

## **Elements of Risk**

*Risk*, or technically *Risk Exposure*, is the interaction of risk's two elements: *impact* if the problem occurs *times* the *likelihood* that it will occur. It's common to portray the interaction of these elements graphically in a Risk Matrix, such as shown in Figure 1.

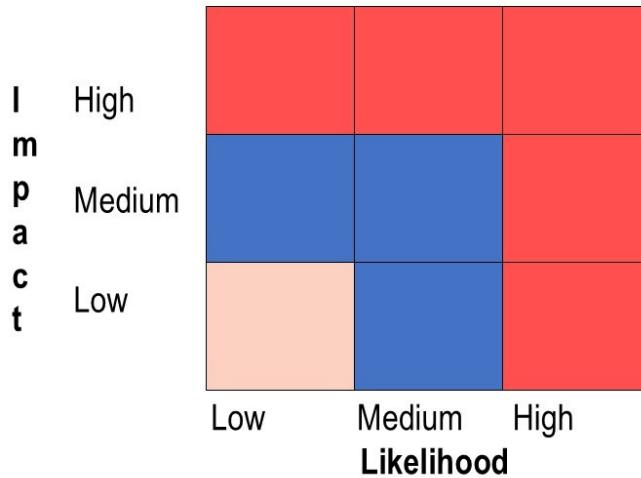


Figure 1. Risk Matrix

Impact increases the more severe damage is, which often is a function of the number of people affected. Sometimes an impact may not be inherently severe but can cause severe consequences. The software industry tends to pay special attention to impact regarding the cost, time, effort, and ability to fix a problem that has occurred. The availability of workarounds can reduce impact.

Likelihood of a problem increases with size and complexity. Technology which is new to the world is more likely to have problems than technology which has been around long enough for many of its problems to have been detected and fixed. Even more mature technology is still more likely to have problems when it is new to an organization which has not yet had a chance to become familiar with the technology. Products and people which have had problems previously are more likely to have (often similar or related) problems. Workers who lack suitable skills, motivation, and methods are more likely to have problems.

Software testing traditionally examines the risks of product *features* and *components*. Features typically tie to key functions the product performs. Feature risk sometimes is referred to as "requirements risk" and frequently is identified as a system's menu

choices. Components relate to how the software is built, that is the technical pieces implementing the functionality. The impact times likelihood of each identified feature and component risk ordinarily is determined in order to prioritize the risks so *the highest risks can be tested most.*

### **Qualitative vs. Quantitative Ratings**

*Qualitatively* rating (typically as high, medium, or low) each individual risk's impact and likelihood is probably the most common risk analysis technique. It's also common to depict these ratings graphically in a Risk Matrix as shown in Figure 1. The graphical matrix can aid communication of findings but does not help the fundamental difficulty of determining degrees of impacts and likelihood.

Many organizations maintain checklists of common risks, which often are based on some of the many risk checklists found on the Internet. Such checklists can aid identifying risks that otherwise might be overlooked. However, over-relying on checklists can interfere with identifying risks specific to one's project.

Moreover, such checklists seldom provide good guidance for assessing impact and likelihood of those risks. Instead, risk analysis usually relies on subjective judgment by individuals with varying levels of relevant knowledge. In additionr, the reliability of such judgments can be further limited when done implicitly, frequently even unconsciously.

In general, *quantitative* measurements are considered preferable to qualitative ones. Quantitative measures are in numbers, as opposed to seemingly less precise qualitative measures such as high, medium, and low. Would you agree that Figure 2 seems more precise than Figure 1?

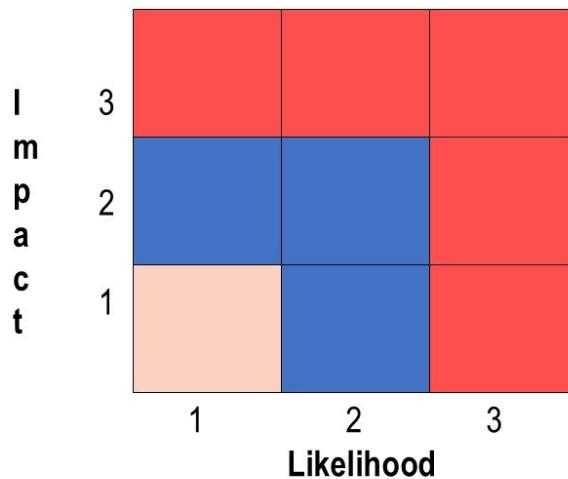


Figure 2. Quantitative Risk Matrix

Note that for the most part, the distinction between qualitative and quantitative risk ratings is fairly illusory. It's illusory because the typical approach to quantification is to make a qualitative judgment and then "quantify" it by assigning a score for each qualitative value. Typical scoring is 1 for low, 2 for medium, and 3 for high impact or likelihood.

Risk exposure, or simply risk, is calculated by multiplying the impact score times the likelihood score. For this, quantifying has an actual advantage, because it provides an easily expressed and understood measure of risk exposure. Thus, multiplying ratings of 1-3 gives a range of risk exposure scores from 1 through 9. A score of 9 is clearly higher risk than a score of 1, 2, 3, 4, or 6.

In contrast, describing the same degrees of risk exposure qualitatively is more cumbersome and less-readily understood. It's harder to recognize the relative risk exposures when stated as low-low, low-medium, low-high, medium-medium, medium-high, and high-high.

### Additional Risk Matrix Considerations

As shown in Figure 3, it's common to shade the risk matrix boxes to indicate risk exposure. Typically, the high-high box in the upper right-hand corner is shaded red. The low-low box in the lower left-hand corner is shaded green. The remaining boxes in

the middle of the diagram are shaded yellow. A common variation would also shade the high-medium boxes red and the low-medium boxes green.

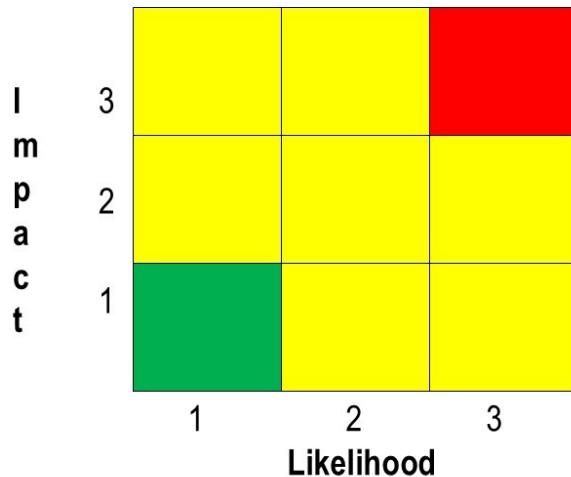


Figure 3. Highlighting Risk Exposure

Such a simple visual picture of risk is very appealing and can be useful. However, caution, it is subject to disingenuous manipulation. Consider what happens to the diagram when the scoring is changed to low = 1, medium = 3, and high = 5. Now the diagram as shown in Figure 4 is a 25-box five-by-five matrix, which graphically implies that a high-high score of 25 is higher risk than an actually identical high-high risk score of 9 on a three-by-three matrix. The effect is even greater when a ten-by-ten scoring matrix is used.

ffffffffffffffffffff

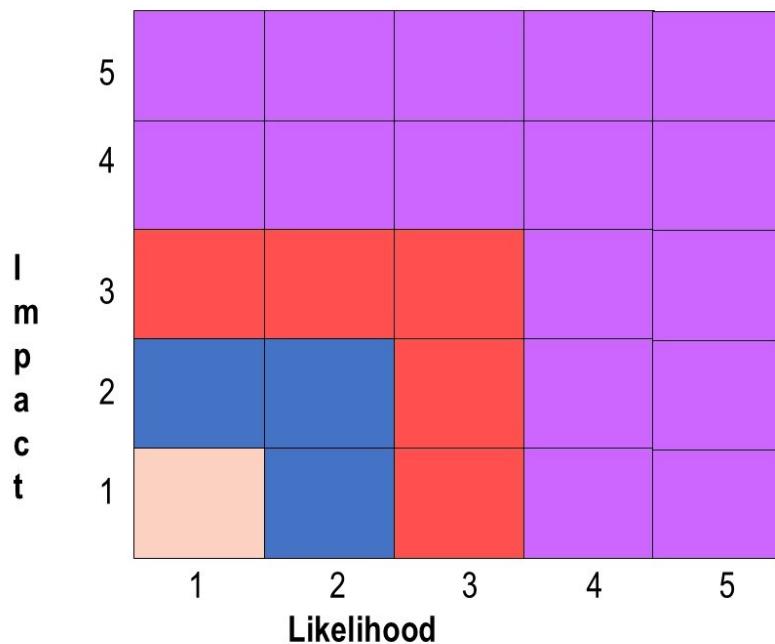


Figure 4. Varying Risk Matrix Appearance

As an aside, a useful but also possibly deceptive scoring approach is to make the scores non-linear. Usually this is done to increase emphasis of very bad outcomes. For instance, the scores might be low = 1, medium = 3, and high = 10 or even 100. In this way, a truly unacceptable riskxz, such as global thermonuclear war, would easily stand out visibly from less severe outcomes, even when likelihood is low.

Another common questionable use of these methods is to represent the truly-artificial “score” as a percentage, rather than arbitrary “points.” Especially when using a ten-by-ten scoring matrix with possible score of 100, depicting risk exposure as a percentage makes it seem more accurate, when in fact such “percentages” are not real. In contrast, some sophisticated organizations, such as insurance companies and airlines, truly use quantitative and percentage measures of risk, which are calculated reliably based on large amounts of relevant data. Regardless, people tend to place greater confidence in risk representations where numbers are assigned as opposed to qualitative scores. It can be valuable to define objective characteristics indicating the respective high, medium, and low levels of impact and likelihood. Such objective scoring helps reduce variability of judgments among raters and thereby enhances risk analysis reliability and repeatability. For example:

Low	= cosmetic,	interfaces with 0 other systems
Medium	= impedes working,	interfaces with 1-2 other systems
High	= won't work,	interfaces with 3 or more other systems

### Rating vs. Ranking

Prioritization is based on analyzing the amount of risk exposure per risk item. The most common method used for determining amount of risk is *rating* and multiplying the impact times likelihood. As discussed above, rating is inherently unreliable when it's done subjectively without objective guidelines. Even when done objectively, though, rating has a number of additional issues limiting its usefulness.

Looking at each choice one-at-a-time takes a lot of time and effort. Even with objective criteria, ratings are often inconsistent and not repeatable, in part because people's evaluations tend to shift based on items they've already evaluated. That is, context matters; evaluating in a different sequence can produce different ratings for the same items.

A bigger issue with rating multiple items is that it often fails to indicate priority. It's common for every item, except perhaps one or two, to be rated high impact-high likelihood. Prioritization requires a way to say an item is higher risk than the remaining items, and so forth; but there's no way to distinguish respective priority sequence of items with equal risk ratings.

Effective prioritization requires forcing distinctions, which is what *ranking* does. Note, Agile's "relative sizing" is essentially a ranking technique. When estimating the "size" of work needed to implement a user story, Agile defines size with respect to the size of another, presumably known, user story. Agile tends not to estimate risk separately but could do so using the same relative approach.

However, ranking also has issues. Continually taking multiple items into account tends to seem harder and take longer than rating each item one-at-a-time. However,, that seeming difference in effort evaporates when one recognizes how much frankly unnecessary additional effort is needed to prioritize equally-rated items.

For ranking to be practical, the number of choices to be ranked needs to be limited to people's span of attention, often referred to as "the magic number seven plus or minus two." If there are too many choices to be ranked, they need to be grouped to enable comparison.

### **Other Confusions Regarding Risk**

Most of what is written and taught about risk deals with *project management risk*. That mainly involves lack of suitable time and resources. While these indeed are relevant to performing testing and may relate to the total amount of testing needed, they do not identify specifically what to test or distinguish priorities for testing more and earlier.

It's common for organizations to rely on checklists to identify risk items.

However, checklists tend to focus on *causes* and triggers of risk occurrences. Causes are what can be addressed to mitigate risks, such as by testing, but are not a suitable

basis for determining risk priorities. Rather, priorities must be based on comparing **effects**—impact times likelihood risk exposure. Very often those identifying risk causes fail to then determine their respective effects which are needed for meaningful prioritization.

### **Reactive vs. Proactive Risk Analysis**

Typical testing risk analysis is *reactive* and relatively weak. Reactive risk analysis tends to come late in the development process, usually reacting to code which already has been written. In the limited time usually available for testing, testers

- Create as many test cases as they can think of
- Prioritize those test cases based on risks they address
- Run the highest-priority tests first and more until they run out of time.

Such traditional reactive testing does catch defects, largely because the development process makes so many; but it also usually misses many of those defects. Reactive testing is expensive, in part because effort often is expended creating test cases that end up not being run. The bigger expense of reactive testing relates to the effects of *risks that it misses* and must be addressed later, when they're harder and costlier to fix.

Instead of starting with time-consuming creation of test cases, each of which deals with a small risk, *proactive testing* looks first at identifying and prioritizing *large risks*. Special techniques which are beyond the scope of this article help identify large risks that ordinarily are overlooked. For the highest priority large risks, similar techniques are applied to identify and prioritize *medium-sized risks*, again including many that ordinarily would be overlooked.

In turn, for the highest-priority medium-sized risks, similar techniques are applied to identify *small risks*. *Small risks are addressed by individual test cases*. Thus, proactive testing only spends the most time-consuming part—creating test cases—on ones that will be run and address the truly highest priority risks. Moreover, this approach prioritizes far more effectively because it considers many ordinarily-overlooked risks.

Proactive Testing's benefits increase further because proactive risk analysis can be performed before a line of code is written. As such, development can avoid many of the design errors that typically produce defects. Moreover, proactive testing enables shifting the sequence of development to build the highest risk pieces first, so they can be tested to catch remaining fewer problems earlier, when they're easier and cheaper to fix. Many of those prevented and early-caught defects otherwise would have necessitated especially time- and effort-consuming redesign and recoding. The amount of avoided work becomes even larger when prevented and early-detected defects also would have affected other pieces of code.

Instead of writing related code and then having to redesign and rewrite it when risks come true, related code isn't written until Proactive Testing risk analysis has shaken out

the design. In these ways, by *letting testing drive development*, risk-based Proactive Testing enables delivering better software quicker and cheaper.

# Revolutionizing Test Automation: Harnessing Generative AI to Empower Product Management and Accelerate SDLC

**Artem Golubev**

artem.golubev@testrigor.com

## Abstract

Test automation has existed for a long time, but are we reaping its benefits to the fullest? Do you think your substantial efforts in script creation, maintenance, and upkeep fail to fulfill the real purpose of test automation?

Time is money, and quality is the best business plan. Today, in the era of DevOps, Agile, AI, and, more specifically, generative AI, you have intelligent and better options available for test automation. In this paper, we will discuss the capabilities of generative AI, which lets you streamline and simplify test creation, execution, and maintenance. The power of generative AI lies in its simplicity and the inclusion of everyone on the team, regardless of their technical competency. This helps you significantly reduce manual effort and time-to-market and enhance efficiency drastically. We will discuss examples and case studies that demonstrate these benefits in visible numbers.

Artem will analyze traditional and advanced technologies, methodologies, and future trends in test automation, highlighting where we are and where we could be. He will also discuss the role of product management in this new era, mentioning how AI tools can empower the leadership to make more informed decisions, optimize resource allocation, and introduce innovation.

This paper explains the transformative potential of generative AI in test automation and quality. We will review what generative AI can accomplish, focusing on its impact on product management and the Software Development Life Cycle (SDLC) acceleration.

## Biography

*Artem Golubev is a co-founder of testRigor, a YC company. He is excited about helping companies to become an order of magnitude more effective in QA and deliver software faster with the help of Generative AI.*

*Artem started his career 25 years ago by building software for logistics companies. Since then, he worked at companies like Microsoft and Salesforce, where he learned about best practices and top technologies in QA.*

*It was a frustrating experience to struggle with building automation and, especially, test maintenance at almost any company he worked for. This is how testRigor was born. testRigor AI now empowers companies like Netflix, Cisco, Burger King, and many more to build test automation faster and spend significantly less time on test maintenance by employing Generative AI.*

# 1 Introduction

Manual testing is suitable for ad-hoc testing, but for the repetitive, complex, and fast test execution, manual testing is not the correct answer. With fast DevOps delivery cycles, manual testing has become a bottleneck, not an enabler. Similarly, the traditional approach to test automation involves significant manual effort in script creation, maintenance, and upkeep. This process can be time-consuming and prone to errors, reducing test automation's overall efficiency and effectiveness. As the software industry has evolved, more intelligent and efficient testing methods are urgently required today. Here, we will see the issues with the current software testing paradigm and how to improve this situation with advanced AI features and capabilities for better quality, efficiency, cost, and timelines.

## 2 Current QA Situation

Currently, you might be using manual testing, traditional test automation tools, or falsely marketed “codeless test automation tools” to meet your testing requirements. The question is, are they delivering how you want them to? Are they helping you achieve the required quality with less manual effort and time? Or are you stuck in expensive production issues, missed deadlines, and never-ending script maintenance? Let us discuss the drawbacks of traditional testing methods.

### 2.1 Drawbacks of Manual Testing

- Bugs slip into the production
- Not enough test coverage
- Time-consuming and error-prone
- Longer release cycles hinder DevOps principles

### 2.2 Drawbacks of Traditional Test Automation (Selenium/Appium)

- Too much maintenance effort, cost, and time
- 5% of tests always break and are unstable
- Slow test creation due to programming
- ROI is low due to the extensive effort involved

Here is a simple login test in Selenium. It requires all the imports, drivers, properties, and dependencies in place before you run the actual test case. Also, as we can clearly see, it is dependent on fragile XPath locators.

```

package tests;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.Test;

public class TestSample {
    @Test
    public void login() {
        System.setProperty("webdriver.chrome.driver", "path of driver");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.browserstack.com/users/sign_in");
        WebElement username = driver.findElement(By.id("user_email_Login"));
        WebElement password = driver.findElement(By.id("user_password"));
        WebElement login = driver.findElement(By.name("commit"));
        username.sendKeys("abc@gmail.com");
        password.sendKeys("your_password");
        login.click();

        WebElement welcomeXPath = driver.findElement(
            By.xpath("//*[@id='body-content']/div[2]/header[2]/div/nav/ul/li[5]/a/span"));
        String welcomeXPathText = welcomeXPath.getText();
        Assert.assertEquals(welcomeXPathText, "Welcome Peter");
    }
}

```

Any change in the element's attributes will cause the test script to become invalid. Quite evidently, you require programming expertise to write this test case. Here is the same test case written through generative AI:

```

login
check that page contains "Welcome Peter"

```

As you can see, the simplicity of generative AI lets you use just a few straightforward English words instead of the whole chunk of programming code. Generative AI enables you to test complex test scenarios with simplicity and ease. You can save the AUT's login credentials in the Gen AI tool and then use the command 'login' every time you want your test script to simulate user login. On average, you can create 5 test cases in a day using generative AI compared to 0.5 using traditional test automation tools such as Selenium.

Additionally, most codeless test automation tools in the market still work on Selenium or programming scripts, and hence, they inherit the drawbacks of traditional test automation tools.

### 3 Role of Generative AI in Test Automation

According to Gartner, by 2026, more than 80% of organizations will use generative AI applications or its application programming interfaces (APIs). AI has been a prominent topic in technology for the last ten years. When generative AI, especially ChatGPT, came out in 2022, it caused worldwide attention and a massive surge in AI innovation and use.

Let us understand: **What is Generative AI?**

Generative AI refers to a category of artificial intelligence that is designed to generate new content or data that is similar to what it has been trained on. Unlike traditional AI models, which typically focus on recognizing patterns or making decisions based on existing data, generative AI creates new, original outputs. This can include text, images, music, synthetic data, and even code.

#### 3.1 Benefits of Using Generative AI in Software Testing

Generative AI offers a modern and intelligent approach to creating software applications and test automation. It uses advanced machine learning algorithms to generate, execute, and maintain test scripts automatically. You can write test cases in plain English or any other natural language to test complex test

scenarios easily and quickly. This approach reduces the dependency on manual scripting and allows for continuous (24x7, in parallel) and efficient testing throughout the SDLC.

- **Write test cases in plain English:** If you choose to write test cases yourself, then generative AI with NLP lets you do that in plain English or any other natural language. Here is an example of selecting a t-shirt size and scrolling the page until the “Pay Now” button is visible on the UI:

```
click "t-shirt"
select 1st option from "Size"
click "Add to cart"
scroll down until page contains "Pay Now"
```

- **Automated test case creation:** You can quickly create automated tests using generative AI capabilities. Generative AI can take descriptions in plain English of what needs to be tested and then identify the key actions and conditions involved.

Provide the test case title/description or describe the feature for which you need the test cases. The intelligent generative AI engine will automatically generate the test steps.

For example, if a tester describes an application (bestbuy.com) as “an e-commerce website where people can search and shop online products”. Then Gen AI will generate automated test cases such as:

1. Order Confirmation and Payment Processing
2. Product Search, Selection, and Checkout Process
3. User Registration and Account Creation Process

It is intelligent enough to then create the test steps for the test case “Product Search, Selection, and Checkout Process” automatically in plain English as:

```
enter "Laptop" into "What can we help you find today?"
click "submit search"
click "Dell Inspiron 15 Touch Screen Laptop - Intel Core i5"
click "ADD TO CART"
```

These steps are generated automatically by the AI without needing the tester to write any code, even the test data is identified by generative AI automatically. Within minutes, you have an automated test case ready to run without any coding hassle.

- **Record and playback features:** The generative AI-based automation tools also allow for record and playback capabilities if you wish to do so. The test steps will be in plain English, and you will have automated tests ready as you navigate through the user flow. For example, creating tests with our Chrome extension will take 4X less time than manual testing; our client IDT’s manual QA team was able to get from 34% to 91% automation in under nine months.
- **Near zero test maintenance effort:** Instead of relying on fixed locators like XPath or CSS selectors, generative AI uses a more abstract approach, often based on the visual and functional attributes of elements. This means that even if the UI structure changes slightly, the AI can still identify the correct elements based on how they appear or behave, reducing the need for

frequent updates to locators. Generative AI has the ability to understand the context in which elements appear.

For instance, if a button labeled "Login" is changed to "Sign In", the AI can recognize that these labels serve the same function and adjust the test case accordingly. This reduces the likelihood of test failures due to minor text or layout changes. As the application under test evolves, generative AI updates its understanding of the application's UI, which allows it to adjust the test scripts automatically. This self-learning capability means that the test cases are in sync with the latest version of the software.

Generative AI uses advanced algorithms to recognize and interact with elements on a user interface, even if their properties change. For example, if the ID or label of a button changes in the application, traditional test scripts might fail because they rely on specific identifiers (XPath/CSS). However, generative AI can adapt by understanding the context and purpose of the element, allowing it to continue interacting with the correct button without needing manual updates to the test script. This lack of XPath and CSS dependency ensures ultra-stable tests that are easy to maintain.

For example, you can support only 200 test cases in Selenium before being stuck 100% at test maintenance. This number is 200,000 test cases for testRigor. Also, our numbers show that it takes 99.5% less time to maintain tests in testRigor than in Selenium.

- **Reusable subroutines:** You can create a subroutine in plain English for a group of test steps, which are often repeated in the test suite. Use the subroutine's name in the test case when you want to run these repetitive steps. Also, if there are changes, you can rewrite the subroutine and need not update each test case separately. Here is a Salesforce test case example of creating a new task (in the Campaigns module):

```

login

create campaign

click "Campaigns"

click stored value "generatedCampaignName"

click "New Task"

generate unique name, then enter into "Subject" and save as
"generatedSubject"

click "Due Date"

click "Today"

click "Save"

check that page contains stored value "generatedSubject" roughly below
"Activity"

```

As you can see, 'create campaign' is a reusable subroutine used in all campaign-related end-to-end test cases. Here are the test steps of the 'create campaign' reusable subroutine:

```

click "Campaigns"

click "New"

```

```

generate unique name, then enter into "Campaign Name" and save as
"generatedCampaignName"

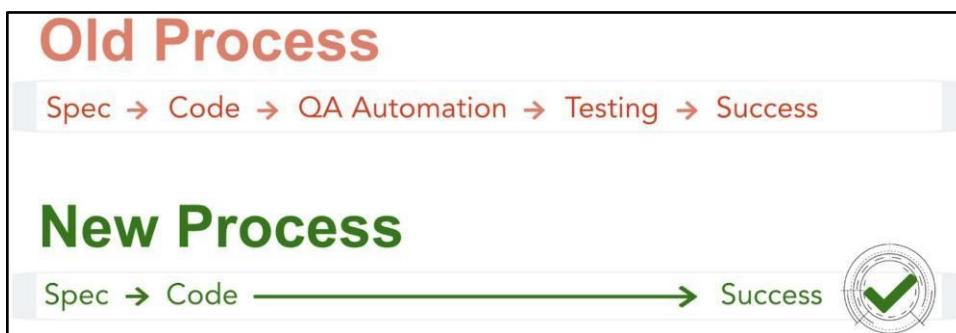
generate unique name, then enter into "Description" and save as
"generatedCampaignDescription"

click "Save"

check that page contains stored value "generatedCampaignName"

```

- **Cost and time reduction:** Automation reduces the need for extensive manual testing, leading to labor and resource savings. Since test cases can be generated quickly in plain English, time that was previously diverted to test script programming is now saved. Additionally, the early detection of defects reduces the cost associated with fixing bugs later in the SDLC.
- **Inclusion of everyone in testing activities through BDD (Behavior-driven Development) and SDD (Specification-driven Development):** Use the power and advantages of shift left testing with generative AI. Create test cases early, even before engineers start working on application code using SDD. With intelligent test automation, BDD is used in its true sense, which is an attempt to express end-to-end tests from an end user's perspective. With plain English test cases, you can write test cases as the end-user will see them without any implementation (programming) overhead. The application code and automation test cases can be generated using the same product specifications.



- **Consistency and stability of test results:** The test results are stable and consistent because there is no reliance on details of implementation (CSS/XPath locators). The non-reliance on basic implementation details makes the test scripts robust and ultra-stable.
- **Empowers manual testers to create robust test automation:** With the power of generative AI, you need not hire separate automation engineering to build test automation. Your manual testers have the right testing skills to test complex test scenarios, which will otherwise take much coding and error handling.

Here are some English commands to accomplish complex test scenarios:

1. **Database:** run sql query "select top 1 UserID, LastName, FirstName  
from Users;"
2. **File:** check that file "instruction.pdf" was downloaded
3. **Email:** send email to "user@customer.com" with subject "Test  
message" and body "Hi, this is a test, this is just a test  
message."

4. **SMS:** sms to +15344297154 with body "hello"
5. **Phone call:** call "+15344297154" and validate it was picked up
6. **QR code:** scan qr code value from stored value "saved-qr-code" and save as "code"
7. **Captcha resolution:** resolve captcha of type recaptcha
8. **Audio:** check that audio from "my-recording" is "70%" similar to "my-specific-recording"
9. **Visual testing:** compare screen to previous version treating error as "minor"
10. **Tables:** check that table "actions" at row "102" and column "Name" contains "Spock"

- **Improved test coverage and efficiency:** Generative AI allows you to write tests in plain English, and your whole team can create and execute test cases. This leads to more thorough testing, identifying edge cases and potential defects that might otherwise go undetected. It increases test coverage and efficiency tremendously.
- **Test scalability:** Generative AI can handle large-scale testing efforts, making it ideal for projects with extensive test requirements. It can quickly adapt to the increasing complexity and volume of tests needed for modern software applications. It can help you find the new corner cases that might get missed by the human eye. Also, the complex workflows (banking applications), and scenarios such as file handling, SMS and phone call testing, Captcha resolution, etc., are handled by generative AI-based testing in plain English.

Test data management, parallel execution, modular test design, parameterization, continuous integration, and regular maintenance are some of the capabilities that generative AI-based test automation provides. This, in turn, helps achieve test scalability.

- **Single tool to test multiple platforms:** The Gen AI-based test automation tools are versatile enough to test web, mobile (native, hybrid), desktop, API, database, etc. This greatly reduces the overhead of using different testing tools for different platforms or devices.
- **Supports continuous testing:** Generative AI being the latest technology, supports continuous testing practices. It seamlessly integrates with continuous integration/continuous delivery (CI/CD) pipelines through CI/CD tools such as Jenkins, GitLab CI, CircleCI, Buddy, etc. This integration ensures that testing is conducted at every stage of the development cycle. It helps with real-time feedback to the development team to fix bugs and facilitate rapid iterations.

## 4 Use Cases of Generative AI Features

### 4.1 Write automation test cases in plain English and copy-paste manual tests from test management systems to automate directly

You can either write your test scripts using the test case editor in English or the record-and-playback tool to record test cases in plain English. There's also the option to let these intelligent tools use their generative AI feature to create a fully functional test case for you with just a description and URL.

Here is a test case example to provide more clarity:

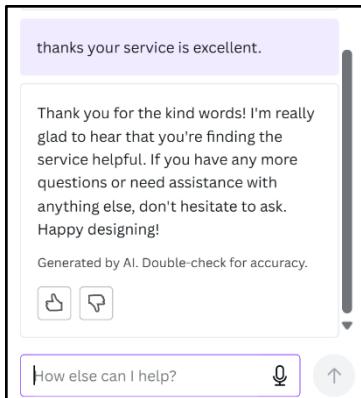
```
enter "Kindle" into "Search"
enter enter
click "Kindle Scribe"
click "Add To Cart"
check that page contains "Item Added To Cart"
```

As you can see in the above example, you can just mention the text visible on the UI, and the elements are identified by AI. Even if the labels are changed, such as “Login” changed to “Sign-in” the AI is able to identify it, and the test case will work flawlessly. Also, the actions and assertions on these elements are in plain English (or any natural language). This is possible because generative AI-based test automation tools use artificial intelligence to reduce the maintenance effort typically associated with automated tests. The reason is simple: traditionally, the automated tests have used XPath/CSS identifiers as locators. Now, with Gen AI, this has changed; the AI uses multiple attributes to recognize an element: context, purpose, behavior, etc. Therefore, it can automatically adjust tests based on application UI changes. This decreases the time spent in maintaining tests after each application update.

You can also import or copy-paste your existing manual test cases from test management tools such as TestRail, PractiTest, Zephyr, etc. Tweak them if necessary, and you can run these test cases directly as automated tests.

### 4.2 Test LLMs and AI features through intelligent AI agents

You can use LLMs (Large Language Models) to analyze and test real-time user sentiments. These findings will help you act immediately based on that information to solve the customer issue and provide greater user satisfaction. Test your LLMs and AI using other LLMs' intelligence. If the real-time user sentiment is not positive (e.g., a user is not able to book a hotel), then you can immediately let customer support take action and aid the user in their user journey. For example, you can verify whether the customer chat has a positive message or not using AI as below:



*check that "chat" "contains a positive message" using ai*

Another example is you can utilize Vision AI to validate a natural language statement about the webpage as below:

```
open url "https://bestbuy.com"
```

```
check that page "has a search bar on the top of the page" using AI
```

All of the above commands invoke AI to analyze the page/screen and perform complex validations that were previously only possible to do manually. Generative AI makes testing advanced AI features such as LLMs straightforward.

## 5 Case Studies: Test Automation with Generative AI

### 5.1 A Fortune 1000 company (IDT) went from 34% automation to 91% automation in under nine months, with only manual QA personnel.

IDT spent 32 person-years of QA Engineers building automated tests. After that, they were not progressing towards their 90% automation goal and were stuck at around 33-34% automation because all of their QA Engineers were 100% busy maintaining existing tests instead of building new ones.

*"We spent so much time with maintenance when using Selenium, and we spend nearly 0 time with maintenance using testRigor"*, says **Keith Powe, VP of Engineering at IDT Corp.**

With generative AI, they achieved the following milestones:

- Turned their manual testers into sophisticated automation engineers with a reduced learning curve when compared to test automation tools based on scripting.
- They are saving a minimum of \$576K worth of costs per year and received 7X ROI
- Since they were using Selenium for test automation, where flaky tests are a real problem, switching to generative AI-based testing reduced considerable time and effort spent on maintaining these Selenium tests. They saved costs by eliminating precious engineering hours spent on maintaining flaky tests or application updates.
- Converted Selenium tests or manual test scripts to stable, easy-to-maintain, plain English-based autonomous tests

They were automating about four test cases per week per person. Now, since they started with generative AI-based testing, they are automating double.

## 5.2 A MicroMobility company (tembici) went from 100% manual testing to 100% automated with an 80% decrease in bugs.

With manual testing, tembici encountered more bugs in production and postponed deliveries. They needed consistent quality environments and use cases where they could execute all their tests at once rather than scattered among multiple developers. Each developer had one device, and they had no concept of a device farm where they could test multiple devices and configurations.

*"testRigor is an easier platform to achieve the consistency of maintaining and running the tests in a speedy way."*, says **Carolina V., IT Management at tembici**.

With generative AI, they achieved the following milestones:

- Developers could begin making tests within an hour of starting testRigor
- Able to test multiple devices without having to write new tests for each device
- Went from testing six builds a week to six builds in just 3 hours
- Eliminated all manual testing

Furthermore, the entire team, from product managers to UI/UX engineers, had a consistent framework for creating, communicating, and executing automated tests. This ultimately led to a more productive team utilizing a tool every member could use.

## Conclusion

By simplifying the creation, execution, and maintenance of test cases, generative AI not only addresses the limitations of traditional testing methods but also enhances the overall efficiency and effectiveness of the Software Development Life Cycle (SDLC). This technology empowers teams across all levels, whether they are seasoned automation engineers or manual testers, to contribute to the testing process with minimal training and technical expertise.

The benefits are clear: reduced maintenance efforts, faster time-to-market, improved test coverage, and significant cost savings. Case studies from companies like IDT and tembici depict the transformative impact of generative AI, showcasing dramatic improvements in test automation levels, cost efficiency, and product quality.

As we progress, product management will continue using these AI-driven tools to optimize resources, make informed decisions, and implement innovation. Generative AI is not just a tool for today but a cornerstone for the future of software testing. It successfully enables teams to meet rapid development cycle demands and consistently deliver high-quality products.

# IoT in the Matrix: Mocking External Dependencies for IoT Systems Using Physical Hardware

**Bartholomew Hsu, Jovan Araiza, Jeff Borg**

[bart.hsu@outlook.com](mailto:bart.hsu@outlook.com), [araiza1852@gmail.com](mailto:araiza1852@gmail.com), [jeff.borg8@gmail.com](mailto:jeff.borg8@gmail.com)

## Abstract

Using software mocks as stand-ins for external dependencies is a well-established practice for software testing. However, mocking subsystems in IoT (Internet of Things) devices that interface with the physical world often isn't practical because the behavior of those subsystems is too complex or opaque. This often necessitates resorting to use of lab test instruments by specialists or field testing with real hardware, which can be difficult and expensive. However, quite a lot of advancements have occurred in hobbyist test instruments, electronics modules, and open-source software over the past couple of years; so that with a little bit of hardware savvy and not too much cost, simulating some of these interfaces has become viable for teams with limited resources. Our team has been able to apply this, using COTS (**commercial-off-the-shelf**) hardware to mock most of the physical-world interfaces of a IoT device such that it's unaware that it's not connected to reality. In this talk, we cover some of the background hardware knowledge necessary, how it can be applied to simulating some of the more complex interfaces, e.g., Wi-fi, global navigation satellite constellations like GPS, power sources etc., costs, advantages and limitations of going with hardware-based mocking, and some of the difficulties we encountered along the way.

## Biography

*Bartholomew Hsu is an experienced tester who specializes in IoT embedded devices and network services. He is currently with Impinj Inc. performing testing for UHF RFID readers and gateways. He received his education in computer engineering from the University of Michigan.*

*Jovan Araiza is a software engineer with 4 years of experience testing IoT software and hardware at Impinj. He has experience testing every layer from web UI down to embedded microcontrollers in IoT devices. He graduated with a bachelor's degree in computer science from Washington State University in 2019.*

*Jeff Borg currently works as an SDET testing IoT firmware. His bachelors' degree (1994) is in Electrical Engineering and his experience includes 20+ years of board level circuit design, including RF circuits. Combining this with significant software coursework at University of Washington, Jeff contributes in an environment where both disciplines are useful.*

Copyright Bartholomew Hsu, Jovan Araiza, Jeff Borg 2024

# 1 Introduction

IoT (Internet of Things) and embedded / mobile devices have subsystems that interface with the physical world, making them difficult for software testers to work with. Simple mocks do not adequately capture behavior and simulating inputs to these systems conventionally requires expensive electronic test instruments and specialists with knowledge to operate them. Software testers were faced with being unable to adequately cover device behaviors that relate to these subsystems.

But that's much less true today than it was ten years ago. Advancements in electronics have resulted in test instruments and electronic equipment that are nearly as capable as their professional equivalents but priced within reach of hobbyists and enthusiasts. Combined with this, a resurgence of public interest in hobbyist electronics and radio has in turn resulted in an increase in both documentation / tutorials and software tools that make use of these devices.

While a bit of self-study and trial and error may be involved, it is within reach of software testers to make use of these resources. Testers can set up hardware and software to mock out these interfaces with enough capabilities to be able to cover important test cases that weren't practical to cover before and at low enough cost to be acceptable. This approach is different from what electrical engineers might suggest but their context and the equipment available to them are much different from that of a software test team.

Our goal is to provide examples that can be set up locally or at least provide insight into the right questions to ask of your colleagues who work with hardware to help you solve the testing problems that you face.

# 2 Wi-Fi Mocking

## 2.1 Context And Motivation

Wi-fi needs barely any introduction, having become ubiquitous as the networking mechanism for IoT / embedded systems where wired networking is unavailable. Our interest as testers is to verify interoperability for devices with wi-fi networks at a functional level. However, even without digging into the RF aspects, wi-fi has many configuration settings, each with its own boundary conditions to probe: SSIDs (which may be ASCII, UTF-8 or just random bytes and hidden or not), PSKs/passphrases, authentication methods like WPA-PSK or the various flavors of WPA-EAP, frequency ranges, and so forth.

Finding a way to do this testing manageably is challenging. Manually reconfiguring a consumer-grade wi-fi access point for scenarios would be time consuming and tedious. Business-grade access points are costly and often come with SaaS for remote management that adds to the cost and complexity. In either case, having a wi-fi access point maintained by the QA or development teams that is connected to the corporate network can raise significant security concerns as well as concerns about wi-fi congestion from the IT department. One might attempt to ask their IT department to provide various configured wi-fi networks instead, but they are not really equipped to provide large numbers of wi-fi networks and would still have the security and congestion concerns.

What we're looking for is a solution that avoids these issues while retaining the capability to freely change the configuration. We can do that by looking for a solution that's somewhat off the beaten path. By using stock Linux packages that implement the functionality of a wi-fi access point (not everything but sufficient for software testing purposes) and some application of basic radio frequency (RF) principles, we can configure a Raspberry Pi 4 with a USB wi-fi adapter to act as a simple wi-fi access point. This then can be directly connected via an RF cable to the wi-fi antenna port of the device we are testing. Creating a wi-fi network for the most common configurations can be done on the fly by revising the settings and restarting the appropriate Linux service. Security concerns are reduced because the direct RF cable connection drastically shrinks the range at which an attacker can attempt to make a connection to the test wi-fi

network, wi-fi congestion concerns are also reduced for the same reason, and cost is low because of the off-the-shelf parts used.

## 2.2 Hardware

Before diving in, we need to digress briefly to explain some basics of working with radio signals which, in turn, explain how to make hardware choices that achieve the desired goal. What we seek is:

- The right combination of cabling and connectors to create a cabled RF connection to pass wi-fi signals between the test wi-fi access point (WAP) and the device under test (DUT).
- To pass the correct range of frequencies between the test WAP and DUT. This is primarily a matter of identifying the frequencies that we are working with and then selecting components that are rated by the manufacturer to behave correctly at the frequencies that we need.
- To pass the wi-fi signal at the correct power level between the test WAP and DUT. A signal at a power level that is too high will damage the receiver circuitry on the other end and can be remedied by reducing the strength of the signal using a device referred to as a RF attenuator. A signal that is at a power level that is too low will not be detected by the receiver and can be remedied by amplification. The goal is to strike a happy medium where the signal strength is just barely powerful to ensure good connectivity.
- To be aware of RF leakage. Much like deep bass notes in music penetrating a wall, radio waves can be difficult to keep contained. Even with the use of a RF cable to contain the wi-fi signal, you may find that the wi-fi network provided by the test WAP is visible from nearby laptops and phones. This may require you to relocate your test rig to prevent attempts at unauthorized wi-fi access.

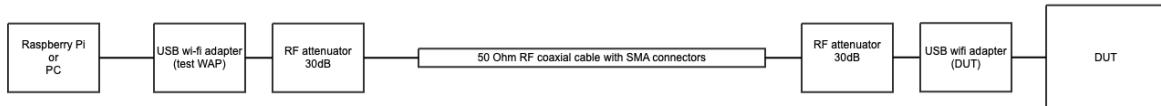
To accomplish those things:

- From the test WAP side, the need to create a cabled connection means selecting a USB wi-fi adapter that has a threaded external connector (RP-SMA) for an antenna to which a cable can be attached. Similarly, from the DUT side, a corresponding means of making an external connection is needed. Many IoT/embedded devices either directly provide an RP-SMA connector or a tiny u.FL connector for wi-fi. For our scenario, we were fortunate that wi-fi support for our DUT was through USB wi-fi adapters with certain chipsets and we were able to procure models that had the RP-SMA connector. Because the components below use normal SMA connectors, a pair of RP-SMA to SMA adapters will also be needed.
  - As an option of last resort for DUTs that do not have an external antenna connector, one might consider asking for a DUT to be modified to replace its onboard antenna with a connector, either u.FL or RP-SMA.
- Identifying the frequency range can be done in a few moments. Wikipedia tells us that the frequency ranges used for the most common wi-fi protocol versions (802.11n, -ac, and -ax) are 2.401-2.495 GHz and 5.150-5.895 GHz. For the relatively short distances (<1m) and the moderate (<6GHz) frequencies involved, 50Ω cables with SMA connectors are inexpensively available. Other components, such as the attenuators discussed next, should also be chosen that have a rating for up to 6 GHz.
- For managing power levels of the wi-fi signal, if we turn to any convenient search engine, we find that the maximum power output allowed for a WAP is 1 Watt (4 Watts for some channels in some jurisdictions but not the one the authors are in) which translates to 30 dBm (decibels relative to one milliwatt). We also find that the signal strength necessary for receiving a wi-fi signal ranges

from -30 dBm to -70 dBm. This means that, in the worst case, we must attenuate the wi-fi signal by a factor of at least 60 dB ( $10^6$ ) to go from +30 dBm to -30 dBm and still have a safety margin of 40 dB ( $10^4$ ) if the actual transmit power is not at the maximum 30 dBm, which it most likely isn't. As it turns out, 60 dB of attenuation in the form of two 30 dB attenuators rated up to 6 GHz attached at each end of the RF cable is sufficient to maintain good signal level.

- It's also worth noting that RF attenuators have a rating for the power that they can safely dissipate that is separate from their attenuation value. In the context of wi-fi, the common 0.5 W rating is sufficient but when working with stronger signals, an attenuator with an appropriate power rating needs to be chosen to avoid damage from overheating.
- For minimizing RF leakage, the least costly and most effective measure will be to use as short a RF cable as reasonably possible to reduce emissions from the cable. Another option is to restrict the maximum transmit power level allowed for the wi-fi adapter from the OS but not all wi-fi chipset drivers support this.

The resulting hardware will look something like



## 2.3 Software

With the hardware side sorted out, we turn our attention to the software side of our test wireless access point. We are using a Raspberry Pi 4B running the current version of Raspberry Pi OS based on Debian 12 ("Bookworm"). We want the DUT to be accessible and assigned an IP on the same network that the Raspberry Pi acting as our test WAP is on instead of a linked but isolated network, as is the default with consumer wi-fi routers one might find at home. In networking parlance, we want the test WAP to act as a network bridge instead of a router. How might we do this?

The first and simplest of our two options is to use the built in functionality of the Network Manager package, supplied and enabled by Raspberry Pi OS by default, that is intended to allow users to set up a temporary wi-fi hotspot. The steps are provided and supported by Raspberry Pi's official documentation ("Use your Raspberry Pi as a network bridge") so there is no need to describe them here. A limitation is that it supports only WPA-PSK authentication, commonly found on consumer wi-fi routers where a single passphrase is used by all devices.

The latter option allows much more configurability but is also much more complex: hostapd. It is also supported on Raspberry Pi OS and was the recommended mechanism for configuring a Raspberry Pi as a WAP in previous releases before the inclusion of Network Manager. Set-up is a bit more complex: a network bridge with an ethernet interface needs to be created through Network Manager. Network Manager needs to be configured to release control of the network interface corresponding to the USB wi-fi adapter so that it can be controlled by hostapd, and then hostapd needs to be installed and configured. Working out the proper configuration is not simple but, for our efforts, we are rewarded with being able to create configurations that match many more common scenarios, including support for the WPA-EAP authentication and its various encapsulated protocols like EAP-TLS and PEAP-MSCHAPv2, both commonly used on corporate networks, to enable multiple authentication mechanisms on the same network.

## 3 GNSS (“GPS”) Mocking

### 3.1 Context And Motivation

The backbone of modern location, navigation, and timekeeping are the several Global Navigation Satellite Systems (GNSS), constellations of satellites broadcasting signals that can be correlated by a receiver to identify its location with high accuracy. Examples include the Global Positioning System, a.k.a. GPS (US), Galileo (EU), BeiDou (China), and GLONASS (Russia).

Mocking out a GNSS signal might be necessary when:

- The GNSS signals are simply unavailable because they are unable to penetrate the building where the DUT is. This is the case for most office buildings.
- There may be a need to check behaviors related to the device being at a different location than its current physical location.
- There may be a need to check behaviors related to large scale movement of the device.

Some buildings have or can be retrofitted with a rooftop GNSS antenna cabled into the building for timekeeping purposes, e.g. for NTP servers or for GPSDOs (high precision oscillators used for electronics testing that use GNSS timekeeping data to stabilize and increase their accuracy) but such antennas are costly to install if not already present or may be in places not accessible for testing purposes. The simple expedient of physically moving the DUT close to window may not always be an option and only provides a half-view of the sky, affecting accuracy and other behavior. The orthodox solution would be to use specialized test instruments to simulate GNSS signals, including multiple constellations at once and movement. But those devices are incredibly costly and are intended for specialists evaluating receivers at a very deep level.

As before, let's look at something unorthodox: there are hobbyist open-source software packages for use with software defined radios that simulate signals from the United States' GPS satellite constellation. The capabilities are limited but sufficient for a GNSS receiver to get a position fix and therefore allow us to execute test scenarios.

### 3.2 Hardware

With continued drops in the cost of processing power, a new way of working with radio signals was developed: the software defined radio (SDR). At a conceptual level, the circuits in any RF system are mathematically transforming the radio signal in stages to get to a final result. An SDR based system instead treats the signal as digital data and processes it using code, with the RF circuitry serving only to convert a received radio signal to binary data for processing or to convert processed binary data to a signal and transmit it as a radio signal. As costs continued to drop, SDRs capable of both receiving and transmitting signals came in reach of people at a hobbyist / HAM radio level, which means more software became available.

The software package we are using, `gps-sdr-sim`, includes a list of SDR models it can be used with. While neither they nor we endorse any specific device, we will note that more expensive SDRs from makers of test instruments (e.g. the Ettus B200 from National Instruments) come with superior documentation, tooling, and support by software packages. Less expensive hobbyist SDRs cost less than a third of the price but may require more expertise and effort to get working. Which trade off you make will necessarily depend on your circumstances.

(At this point, we are obligated to digress momentarily into legal and regulatory matters. You may see tutorials and video demos where a person hooks up a SDR to an antenna and transmits over-the-air to demonstrate simulating GPS. You may be tempted to do the same if your DUT does not have a GNSS

antenna port. These demos are almost certainly being performed illegally. Readers should be aware that, in nearly every jurisdiction globally that

- transmitting a radio signal requires compliance with regulations and appropriate licensing
- that a signal being transmitted is easy to track down with the right equipment
- that the fines and penalties for those caught can be draconian

In short, never use a SDR to transmit using an antenna unless you are very sure that you are doing so in compliance with the laws of your jurisdiction.)

Going back to our RF basics discussed in the section on wi-fi:

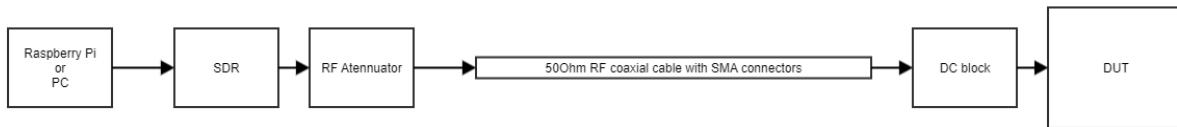
- A brief visit to any search engine reveals that the primary frequency range (the L1 band) used by GPS is centered at 1575.42 MHz with most other GNSS constellations having their L1 frequency either at the same frequency or nearby. While you should still check the frequency ratings for cables and other components, this frequency is low enough that most components should be capable of handling it without any problem.
- Again turning to a search engine, we find that the signal strength that would constitute a sufficient signal is -125 dBm to -155 dBm. (These are fantastically small quantities:  $10^{-15.5}$  to  $10^{-18.5}$  watts which highlights what an engineering marvel GNSS systems are.) This means that we will need attenuators again but how many depends on the signal strength being put out by the SDR. For the moment, let us stipulate that attenuators will be needed and that we will need to take a measurement of the output to determine exactly how much is needed.
- At such low signal levels, leakage is not an issue.

To those, we add one other consideration:

- Because of the extremely weak signals involved, GNSS receivers are often operated with antennas with built-in amplifiers. These amplified antennas require power so the receiver will supply a voltage through the antenna cable to the antenna to power it. Needless to say, a SDR's transmitter output is not designed with the presence of this voltage in mind and there is potential for it to be damaged. Therefore, one additional component is needed: a DC block, which, as its name suggests, will block the unwanted fixed voltage supplied by your GNSS receiver while still letting the RF signal pass through. Forgetting to install the DC block may result in damage to the SDR.

We still need to determine the base output power of a signal transmitted by our SDR of choice so that we can determine how much attenuation to use. To do that, we need to introduce one other test instrument, the spectrum analyzer, which, to oversimplify, when tuned to a particular frequency range, will show how much power in dBm is detected in that frequency range as a graph. (If one is not available, we can again turn to hobbyist grade instruments: the TinySA Ultra is well regarded in HAM radio circles, is very inexpensive, and has been used by us to take such measurements). The procedure is simply to connect the SDR's transmit output to the spectrum analyzer with an appropriate attenuator attached for safety, configure the SDR to transmit a sine wave at the L1 frequency, 1575.42 MHz, with zero amplification added by the SDR, configure the spectrum analyzer to examine the L1 frequency range, and read off the height of the peak, which will be the default dBm power. If the signal is not visible, reduce the size of the safety attenuator and try again. From that value, subtraction and then addition of the dB value of the safety attenuator will tell us how much additional attenuation is needed to get to roughly -125 dBm. Precision is not needed; to within +/- 10 dBm should be fine.

The resulting hardware looks something like



### 3.3 Software

On the software side, as mentioned earlier, we will be using an open-source package named `gps-sdr-sim`, which is available as a repository on GitHub. `gps-sdr-sim` does not operate in real time; instead, it generates a signal file representing the GPS signal and that file is played back through an SDR. To summarize the process:

- Choose the latitude, longitude, and height above sea level you wish to use for your simulated antenna location and specify it as a static location. (The `gps-sdr-sim` does provide support for simulated movement as well but that functionality has not yet been evaluated.)
- Choose a date and time and retrieve the appropriate GPS constellation ephemerides file from the NASA CDDIS site (“Daily RINEX V2 GPS Broadcast Ephemeris Files”). These files describe the location and movement of each of the GPS satellites in orbit, which determines the set of GPS satellites that are visible from the chosen location and what distance they are at. This is necessary to compute the RF signal that the GNSS receiver should be receiving at that location. Because each file covers a timespan of one day, the time from which the simulation can start is limited by the range of the file.
- Choose a duration. For doing GNSS testing, the concept of time to first fix (TTFF) and cold/warm/hot starts should be familiar. The duration chosen should be for the cold start TTFF plus the scenario duration.
- Issue the command to the GNSS receiver to reset to the cold state. Remember that the generated signal file isn’t going to be synchronized with the actual current time. Having the receiver in the cold state ensures that it doesn’t see an unexpected jump in time from any previous runs.
- Issue the command to begin playing back the RF signal file through the SDR.
- Begin observing the output of the GNSS receiver. This is usually through a serial connection exposed to the Linux OS as `/dev/tty*` outputting NMEA 0183 formatted data sentence in plain text. The GPGGV sentence output should begin to show satellites becoming visible as well as their apparent signal strength. You may need to increase the amplification level of the SDR playback command if the signal strength is low or no satellites are visible. After sufficient satellites are visible and at or before the cold start time has elapsed, the GPGGA sentence should be populated with the latitude and longitude that the receiver thinks it’s at. Other NMEA 0183 sentences may be available, refer to the documentation of the receiver module being used.
- At this point, the GNSS receiver module has and should retain a fix until the playback ends and your tests can begin executing.

### 3.4 Learnings And Limitations

There are quite a few limitations:

- As mentioned before, this simulation isn’t synchronized with the actual time. This does not pose an issue for the authors’ test scenarios but may pose a problem for other scenarios.
- Simulation time can be constrained on low end computers like the Raspberry Pi 4 by the large file size of the generated RF signal, roughly 3 GB for 5 minutes worth of signal data, and by computation time, nearly 1 second of CPU time to generate 1 second of signal data.

- This simulation has not been evaluated against receivers with Assisted GPS (A-GPS) enabled. Such systems will have satellite ephemerides data for the current time, restricting the start time for the simulation to near the current time.

These limitations, while considerable, need to be weighed against the cost savings.

## 4 Battery / Power Mocking

### 4.1 Context And Motivation

Maximizing battery life is crucial for mobile devices and battery powered IoT applications and part of that is being able to assess power consumption in varying test scenarios. Some devices, like smartphones / tablets, have a built-in battery fuel gauge chip that can provide this data for you but not every DUT includes this kind of circuitry. In that situation, we can look instead to lower cost power monitoring test instruments instead that will effectively be a mock taking the place of an actual battery.

### 4.2 Hardware + Software

To provide the necessary background in brief, a battery contains a certain quantity of energy that it can deliver, much like fuel in a fuel tank, before becoming fully discharged. That total energy is measured in terms of power delivered over a duration: watt-hours. The rate at which energy is delivered from the battery varies depending on what the DUT is doing, so measuring power consumption boils down to measuring the current and voltage at very short intervals, computing the power from that, and adding together these myriad measurements together to determine the total energy consumed. The measurement frequency affects the final accuracy as well; measurements taken at a high rate are less likely to miss transient spikes in power usage.

Fortunately, test instruments exist that do the job are available at a manageable cost compared to full sized DC benchtop power analyzers. For our test configuration, the Monsoon Solutions High Voltage Power Monitor (Model AAA10), is a combined power supply and high precision power measurement tool that meets the voltage and current requirements for the products we are testing. (For handheld/mobile devices that use batteries that deliver <4.5 V, there is a corresponding Low Voltage Power Monitor (Model FTA22D) that is readily available on the secondhand test instrument market.) The manufacturer provides Python and other libraries for configuring the instrument and taking measurements. For test configurations that have higher power requirements and where therefore less precision is needed, benchtop power supplies with USB/network interfaces that allow power readings to be logged to a computer may be a viable lower cost option.

### 4.3 Testing process

The test process is:

- Configure the voltage to match the requirements of the DUT
- Boot the DUT
- Set the DUT into the desired state for which power measurements are to be made
- Begin logging power measurements through the power measurement tool's API
- Allow the DUT to execute the scenario
- Perform analysis of the gathered data, including computing the average power consumed and checking the data points for unexpected bumps/spikes in power consumption

Possibly the biggest difficulty with making power measurements is repeatability, particularly for handheld devices which necessarily have small batteries and are therefore the most severely power constrained.

Because any modern IoT device is a veritable hive of processes and subsystems with each conducting activity on its own schedule, if you graph power consumption versus time, you will see many transient spikes caused by that activity. It is unfortunately effectively impossible to trace those transient events to the activity that originated it. Adding instrumentation and logging to the code to attempt that would itself trigger additional activity that adds more noise to the data, defeating its own purpose. Ultimately, we need to accept that making power measurements requires us to deal with some level of randomness which requires use of statistical principles to tame. Some tips:

- Putting the DUT into as quiescent state as possible helps improve reproducibility. Disabling services / software and turning off any hardware subsystems not relevant to the scenario being executed can help eliminate noise in the measurement. This has to be balanced against the risk of turning off too many things and causing behavior that doesn't reflect what the end-user will see.
- For average power measurements, increasing the duration of the measurement can reduce the effect of noise on the average. For example, a one-off 1 second burst of activity can significantly skew the average power measured over a 1 minute duration but has very little effect on the average power measured if the measurement duration is instead 1 hour.
- The inverse works as well: one can amplify the effect of an action that individually has very little power consumption, enough to make it measurable, by repeating it many times over a long duration. Assuming that you have a baseline average power measurement that's repeatable, you can take an average power measurement while repeatedly triggering an action N times and then subtract that average power measurement from the baseline average power measurement and divide it by N to get an estimate of the power consumption for that action.

## 5 Mocking other interfaces

### 5.1 Mocking for GPIOs and other digital I/O

Industrial embedded systems and IoT devices often have the capability to interact with external hardware through GPIO (General Purpose Input/Output) pins that can be configured through software to act as either input signals or output signals, e.g. handling buttons or other user inputs or triggering buzzers / lights for the device being controlled.

For testing purposes, probably the most convenient and easily accessible way of mocking out a GPIO interface is the Raspberry Pi, which itself has a set of GPIO pins and well documented and supported libraries to configure them, making it relatively straightforward to mock out any device one would connect to the DUT's GPIO. In the authors' case, the Raspberry Pi's GPIO were configurable to be directly compatible with the DUT, but in some cases, you may need an additional circuit to translate from one interface to another.

### 5.2 Mocking options for RF signals

We've talked about the RF signals for wi-fi and GNSS in preceding sections but there are IoT devices that work with other types of RF signals, e.g. the authors work with UHF RFID tag readers operating in the ISM frequency range. There are a couple of other things that software testers can deploy to manipulate RF signals for mocking purposes: USB RF switches and USB RF attenuators.

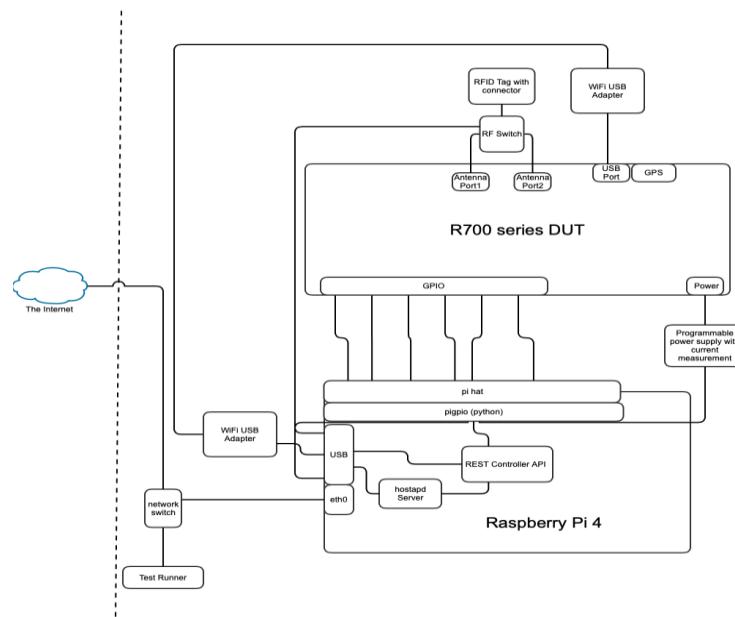
A USB-controlled RF switch is a bi-directional device that can be used to switch a single "common" RF port to connect to one of a set of RF ports by issuing commands to it over USB. The manufacturers provide libraries and sample code to do so, although the quality may be variable. As an example of how it might be used, a number of the authors' test scenarios involve a UHF RFID tag wired up to a SMA connector (instead of its usual antenna) connected to the common port of a USB RF switch and with the remaining switched ports connected to the antenna inputs of the RFID reader. With this arrangement, we

can make the tag seem to be visible from any of the RFID reader's antenna ports by simply issuing a command to the switch.

Though not currently used by the authors in any test scenarios, another device worth being aware of that apply to test scenarios are USB-controlled RF attenuators which provide programmable attenuation to signals passing through the device.

## 6 Overall system

A system diagram is presented below with all of the mocked out interfaces in place. The DUT is operating as it normally would, unaware that its hardware interfaces are not connected to the real world.



To provide control over these mocked out interfaces, we created a management REST API that is served through a Python web server. Tests that are executing can simply make the appropriate REST calls at the appropriate time to set the configuration needed for the current test scenario.

## 7 Conclusion

In conclusion, we hope we have shown that advances in hobbyist-grade electronic tools and less well-known electronic test instruments have put mocking out hardware interfaces for the mobile devices and IoT systems they within the practical reach of software testers. While it is true that a bit of electronics and other knowledge is needed to take advantage of this, most of the knowledge is terminology and various rules of thumb rather than anything complex.

## 8 Disclaimers

The views expressed in this paper and presentation are those of the authors, not Impinj Corporation.

## 9 Acknowledgements

The authors gratefully acknowledge the support provided by Impinj for this paper as well as the support of colleagues there who have provided review and feedback to improve it.

# Trust by Design: Building Credibility in AI with User-Centered Principles

**Zahra Langford**

[zahralala@gmail.com](mailto:zahralala@gmail.com)

## Abstract

The current state of AI products for decision-making in the market is disheartening. As the hype fades, realization sets in that these tools are unreliable, and more likely to create extra work in the short term than reduce it. These products often fail in complex real-life scenarios, are difficult to use, and untrustworthy. As technology creators at an inflection point in our industry, we have an opportunity to slow down and take time to reevaluate the processes used to build AI products so that we can move forward more thoughtfully while earning users' respect and trust.

The goal of this paper is to contribute to the practical understanding of how to build better AI products for decision-making from a user experience (UX) design perspective. It details design principles,

- Center Users – Core Principle: Prioritize user needs at every stage of AI product development.
- Articulate Trustworthiness – Ensure transparency in AI product algorithms and operation.
- Exhibit Confidence – Demonstrate the reliability of AI product outputs.
- Illuminate Logic – Explain the reasoning behind AI product decisions.
- Facilitate Action – Enable users to act effectively on AI-generated output.

It includes visual examples and practical strategies for applying these principles to the AI products you may be working on.

Together, we can impact the usability and trustworthiness of AI products industry wide. We can increase the user-centricity of our products even if we only focus on one or two of these design principles to champion within our organizations. These principles support the creation of products that delight users and make their lives easier, while rebuilding the trust that is so critical to AI technology moving forward.

## Biography

*Zahra Langford is a user experience design leader with 15+ years championing end-to-end design of artificial intelligence (AI), healthcare, finance, and ecommerce products for SaaS B2B and B2C companies. She designed her first machine learning app in 2012 as Principal Interaction Designer at the Xerox Innovation Group. She is known for user empathy, curiosity, and measurable results. She currently works as the UX Program Manager and Designer at Rogue Credit Union. Zahra is also a small business owner; she is always experimenting with AI products to augment her workflows to understand customers and create nature-inspired fashion and home décor designs.*

Copyright Zahra Langford 2024.

# 1 Introduction

The recent explosion of AI products aimed at improving decision-making, combined with the rush to get technology to market has led to unusable products, broken promises, and a deepening lack of trust in AI companies, further compounding trust issues inherent in the field (Chakravorti 2024).

The *AI Gone Wrong* page maintained by tech.com, a news and reviews resource, documents a growing list of AI product errors, mistakes, and failures by prominent technology companies (Drapkin, 2024). Only 15% of product leaders in North America and Europe report that their users are embracing AI features (Pratt, 2024). As we descend into the “trough of disillusionment” of the Gartner Hype Cycle™ for AI (Gartner Research, 2024). There is an opportunity to step back and reassess our process before we alienate users entirely with public missteps, half-truths, and misunderstandings of their needs and wants.

The design, development, and delivery of these products is often missing real connection to users and their goals. Few are asking, “Do users want or need these products and features?” “Are they solving real user problems?” “How can we demonstrate accuracy and confidence?” Let us stop leading with technology and start leading with user-centered principles.

Our primary challenge is to build usable and trustworthy products that meet users' needs while balancing technological advancement and competitive advantage.

# 2 Purpose

This paper aims to support the delivery of AI products for decision-making by providing design principles to ensure these products are desirable, useful, usable, and trustworthy for target-users. It documents the use of design principles to create better AI products and demonstrates how to apply these to real-life products. By conclusion of this paper, you should understand industry challenges, have design principles to address them, and practical ideas for transforming this information into actionable steps to build the next generation of transformational AI products.

# 3 Definitions

## 3.1 Artificial Intelligence (AI)

- AI - Systems capable of performing tasks that normally require human intelligence, consisting of algorithms and models that enable these systems to learn from data, recognize patterns, and make decisions (GeeksforGeeks 2024).
- All AIs referenced in this paper are Narrow AI (designed and trained for specific tasks or domains) expert systems and do not include physical/mechanical systems (GeeksforGeeks 2024).
- The AIs in this context can consist of one or more sub-systems, including Computational Linguistics (CL), Large Language Models (LLM), Machine Learning (ML), and Natural Language Processing (NLP).

## 3.2 AI Products

- AI-driven products - where the product and the AI that powers its features and functionality are built at the same time and the product is not functional without AI.
- AI-assisted products - products that already existed, to which AI-driven features are integrated. Example: Microsoft 365 products integrate Copilot as an AI assistant.

This paper will use the term “AI products” to refer to both AI-driven and AI-assisted products. The focus will be on products that analyze large chunks of structured and unstructured information to generate results such as recommendations, predictions, identification, and summarization that support strategic decision-making.

### **3.3 Design Principles**

Product design principles are value statements that apply to the presentation layer of products, outlining the key objectives that products should deliver for their users. They frame design decisions, typically focusing on UX design fundamentals such as “Provide the ability to undo and redo actions” or “Be flexible,” but also those that align with a specific mission and vision like “Simplicity” and “Inclusion.”

Every organization can benefit from a set of core design principles. In this case, the focus is on a set of principles tailored to the unique challenges of designing and developing AI products.

The principles in this paper are written to integrate into a company’s core design principles. If an organization does not have existing principles, let the principles documented in the following pages inspire you to help create them and incorporate these with an AI perspective. Feel free to customize them, every company has a unique way of articulating principles to match their mission, vision, target users and brand.

## **4 Challenges**

The rapid growth of AI in technology comes with inherent internal challenges to the creation of user-centered and trustworthy products.

- Companies face pressure to make statements about AI capabilities and release products before they are fully developed and tested. Announcing exciting new AI capabilities can boost financials, but failing to deliver as promised can result in decreased stock prices, loss of valuation, lawsuits, and penalties (Donachie 2024).
- Companies often lack incentive to fully disclose what their AI products can and cannot do, clearly explained, stored, and anonymized, and identify the level of accuracy and confidence in results.
- The highly competitive nature of the industry fosters secrecy to protect competitive advantages, limiting transparency about how the technology works.
- There is often a drive to acquire more users to improve product algorithms, with less consideration given to retention and repeat use.
- It may be difficult to explain how the AI product generated a specific output due to its independent learning capabilities.

In addition, pushing technological boundaries while maintaining user focus is difficult. Users often cannot articulate their needs and may not recognize the benefits of certain technologies until they can experiment with them.

“What people say, what people do, and what they say they do are entirely different things.” - Margaret Mead

These challenges can limit our efforts. However, the design principles presented in this paper will aid in overcoming them.

## 5 Design Principles

This collection of design principles for AI decision support products has evolved over the course of 15+ years in collaboration with esteemed data science, design, and engineering colleagues.

### 5.1 Core Principle: Center Users

This key principle should be at the heart of everything. Users come first. We need to understand the target users deeply before applying AI to solve their challenges. Users should guide technology, not the other way around. Simply staying committed to users, their needs and wants, will organically increase the quality of AI products.

Think of users as more than sources of training data, consider the advantages of building long-term relationships with repeat usage that will drive product stability and success. A user-centered foundation can inform the integration of AI functionality that learns about users as they interact with the product, which can automatically evolve a more personalized experience for them over time.

Make centering users a priority with these straightforward strategies to improve user focus and product relevance from inception.

#### 5.1.1 Everyone is not Your User

Remember that “everyone” is not a legitimate target-user population. Identify exactly who you are designing for and what problems you are solving for them before completing major work. This is your **North Star**. Start with something easy and clear, it can evolve as you learn more. Keep it documented and visible as the project progresses. Remember that there may be additional users beyond your primary ones such as admins and specialists. Here are ideas to help you get clear on your target user (or users).

Try to answer these simple questions, use relevant qualitative and quantitative data if available.

- Who are the users? Define characteristics such as demographics, profession, income, device, screen size, and OS.
- What do they want? Understand their challenges, needs, and desires.
- What problems are we solving for them? Explain how the product will improve their lives.
- What does user success look like and how is it measured?

Stakeholders and team members should be able to explain the product to people on the street with limited technical backgrounds. For example: Users X and Y, who [key characteristics], struggle with A and B due to C and D. This product addresses these challenges by [how it improves their lives], enabling them to achieve E and F more effectively, with success measured by G and H.

Build user data collection into products so you can continue to learn more about them and make relevant improvements over time. The user focus does not end with the first deployment, this data will contribute to effective product improvement road mapping.

### **5.1.2 Find Real or Representative Users**

Access to users is critical to understanding, helpful for answering questions, and validating direction. Product team members are not the users of the product they are bringing to life and should not attempt to be. If there is no access to real users, find people in the company who can accurately represent users such as customer support representatives or business analysts. Ask people who are close enough, in demographics and characteristics, to the target user and who are easily accessible. I have found that internal personnel can be immensely helpful because you do not need an NDA, and interactions can be more informal.

### **5.1.3 Address Bias**

This could be a topic for an entirely separate conference, so this paper does not go into detail. The most important thing to keep in mind is that it is never safe to assume that AI algorithms do not contain bias. Approach them as if they have bias built in and the capacity to produce biased output when trained on existing content. Try to do everything you can to reduce this bias, it should be a continuous process (Timbo 2023).

### **5.1.4 Keep it Simple**

Keep it simple stupid (KISS). Refer to your north star when deciding what features and capabilities to include and how to present them in the interface. Make sure the most important things are the easiest to identify and understand. Secondary functionality can be less prominent or a click or two deeper. One of the most common issues with most products is that they try to do too much and crowd everything into a single view.

### **5.1.5 Consider Accessibility**

Consider accessibility before starting the project. Identify Web Content Accessibility Guidelines (WCAG) 2.2 Level A (minimum) requirements and Level AA requirements (widely considered legally acceptable) to integrate into product planning (W3C Web Accessibility Initiative 2024).

## **5.2 Articulate Trustworthiness**

Users should not have to search for the information they need to make decisions about placing trust in AI products. A summary of trust indicators should appear prominently on the product's entry pages and in any descriptions of the product and its capabilities. Include confirmation of trustworthiness at key points of interaction during the product's use (see Design Principles 5.3-5).

AI products on the market can be very vague about what they do. Companies force users to create accounts and experiment with the capabilities to understand if it will help them with their workflows. If users discover it will not, they have already sunk a significant amount of time and money in subscription or access fees. This obfuscation can alienate users, making it unlikely that they will return, even after product teams implement improvements.

When users first encounter the product, they immediately seek indicators that it is trustworthy and can meet their needs. Begin with a clear explanation of your **North Star**: who the product is for, what problems it solves, and what makes it special. Then, present the company's high-level values, articulating their commitment to transparency and ethics, including specific details in clear and easy to understand language. (Deodhar, Borokini and Weber 2024).

- Model definition, use, and behavior.
- Model training data content and source.
- Established industry frameworks incorporated to arrive at correct and usable output.

- Data privacy policy with the ability to opt out of data tracking. Describe how the system collects and utilizes user data.
- Security information describing how the product keeps users' data secure.
- Privacy or security certifications obtained.

### **5.3 Exhibit Confidence**

Assure users of the accuracy of the AI product's output. Here are strategies for demonstrating reliability and confidence.

- Reference supporting sources or industry data and include reputable sources used to generate and validate results.
- Use an established confidence scoring methodology or design a validated one that is clear to non-technical users.
- Clearly label outputs that are speculative and need further validation.
- Provide ways to make independent validation easy.
- Ask users to give feedback on the quality or usefulness of the output.

While researching this paper using Microsoft's Copilot in Bing, the tool gave me data that seemed inaccurate. I prompted it to give me a confidence score for that data. The response: "Certainly! When generating responses, I do not directly provide a confidence score. However, I strive to be accurate and helpful." This was extremely unhelpful and diminished my trust in the output.

Here is an example from my work at HealthPointe solutions as the Director of Product Design (2019-23). This is a page from the Enhanced Medical Chart product that helps medical providers get a useful and searchable summary of their patients' health. It surfaces insights about potential conditions from unstructured notes in the patient records that would otherwise go untreated.

In this scenario, a provider has searched for 'HCC' (Hierarchical Condition Categories: sets of medical codes linked to specific clinical diagnoses) to find any undiagnosed conditions. The product surfaced three suspected HCCs and it gives a confidence level for each. Selecting one of the HCCs links the provider directly to the places in the unstructured medical record where the evidence is located, so the provider can validate for themselves the decide whether to document an official diagnosis and start a treatment plan for that condition or disregard it.

**Search Results**

Julianna Moore (DOB 05/05/1976)

**Contents**

4 Results for 'HCC'

**3 Suspected HCCs**

- Evidence Package for HCC 135**  
HCC 135: Acute Renal Failure  
Code N17.9 Acute kidney failure, unspecified  
Group: Kidney Disease  
Evidence Time Frame: 2021-2022
- Evidence Package for HCC 88**  
HCC 88: Angina Pectoris  
Code I20.9 Angina pectoris, unspecified  
Ischemic Heart Disease  
Evidence Time Frame: 2021-2022
- Evidence Package for HCC 108**  
HCC 108: Vascular Disease  
Code I73.9 Peripheral vascular disease, unspecified  
Group: Peripheral Vascular Diseases  
Evidence Time Frame: 2021-2022

**1 Other HCC**

- Evidence Package for HCC 161**  
HCC 161: Chronic Ulcer of Skin, Except Pressure  
Code: L98.499 Non-pressure chronic ulcer of skin of other sites with unspecified severity  
Group: Diseases of The Skin and Subcutaneous Tissue  
Evidence Time Frame: 2021-2022

Search results from HealthPointe's Enhanced Medical Chart showing undiagnosed conditions by strength of evidence. No real patient data was used to generate these screens. Proprietary. © HealthPointe Solutions. All Rights Reserved.

## 5.4 Illuminate Logic

This principle is an early vision of what is now evolving into "Explainable AI." Trust builds on understanding. Products should reveal how they work by showing the source data and, when applicable, a decision tree used in the final output. This can be difficult as companies must safeguard proprietary methods unless committed to an open-source model. However, there are ways to do this that meet the users' need for understanding while protecting competitive advantage (Majewski, 2022).

- Link to original source material so the user can evaluate its trustworthiness.
- Offer a high-level explanation of key factors used to reach a conclusion. For example, explain that the model predicted X due to factors Y and Z.

Most AI-driven search engines now include links to references in the top-level results summary. However, this does not seem to have made its way into more complex decision support products.

Here is another example from HealthPointe Solutions' Enhanced Medical Chart, linking a code prediction to specific appearances in the patient's medical record. It shows the exact information used to make the recommendation.

**Enhanced Medical Chart**

**Page No: 1** **Juliana Moore (05/05/1976)** **Close**

Contents & Search

**Legend:**  Conditions and Symptoms |  Medications |  Procedures |  Labs |  Allergies |  Life Styles |  Dates |  Other

**ED PIT note - ED Notes (Inpatient Encounter)** **05/16/2020**

PIT Course (ED PIT note)

83 yo male brought in by EMS complaining of possible **GI bleed**. Patient reports he noticed **frank blood** in the toilet tonight. He reportedly passed out 1-2 times. He was just **transfusion**.

On exam patient appears slightly pale. Contusions noted. Workup initiated in triage. I initiated the **medical screening exam** of to begin workup. \*\*\*\* A \*\*\* PA **05/16/2020** 2154

**Discharge Summary - Encounter Notes (Inpatient Enc)** **05/16/2020**

5/22/2020 3:35 PM. Hospitalist Discharge Summary. Patient Date: **5/22/2020**. y.o. Gender: male Admission Date: **5/17/20** Length of Stay: 5 days. Major/New hospital diagnosis: **hypovolemic shock** G. Significant Laboratory results needed for follow up: **CB**. Significant Imaging results needed for follow up: N/A. Significant Pathology results needed for follow up: N/A. Anticoagulation and follow up needs: D/C **coumadin**. **Medication changes** and reason: PPI BID **metoprolol** reduced. New DME: N/A. Consultant recommendations: follow up with heme/onc and GI outpatient. Discharge Disposition: Home or Self Care.

Immunization History (Discharge Summary)

**Information for Condition or Symptom**

MS DRG		
791		

Clinical Codes		
Rank	ICD Code	Code Desc
1	K92.2	Gastrointestinal hemorrhage, unspecified

Appearances: 5		
Section	Pages	
H&P - Encounter Notes ED PIT note - ED Notes	10	11   20 1

< Previous: Vaccinations Go To Page: **1** Next: Page 2 >

Detail of patient record from HealthPointe's Enhanced Medical Chart showing evidence to support its automated condition coding. No real patient data was used to generate these screens. Proprietary. © HealthPointe Solutions. All Rights Reserved.

## 5.5 Facilitate Action

The promise of AI extends beyond accurate and trustworthy output. Empower users to make informed decisions and act on information, driving change, and delivering tangible results within the same ecosystem. AI products often address only one part of a user's workflow. Ideally, users should not have to move information in and out of other systems to complete those workflows. This can be difficult to accomplish and adds more effort, but therein lies the true value of AI.

Financial institutions employ AI products to make automated decisions on loan and credit card applications, reducing effort on the straightforward applications while sending more complex cases to a human for processing. Healthcare coding systems can make automated decisions for billing purposes, using code prediction like the example above, sending invoices to insurance companies with a high rate of success.

Actions can be as simple as sending an email with a report to ask another person to complete the necessary work, such as making a phone call, submitting a referral, or scheduling appointments. If you cannot integrate actions into the product, explain how users can act on the information provided in diverse ways. Include the ability to export data and reports out of the product in a way that is consumable by other systems.

Here is an example from HealthPointe Solutions' Digital Care product that assists medical providers in managing their patient population and improving outcomes for high-risk patients. A provider can select a patient or patient group with similar characteristics such as conditions, risk factors, or appointment

recency. Then the provider can act on the insights or circumstances. Actions in this context are: Offer transportation assistance, assign to social work or nurse to call patient.

The screenshot shows a digital interface for a patient named Julianne Moore. At the top, there is a profile picture and the name "Utilization Gap for Julianne Moore". Below this, a note says "Lab tests are irregular and sporadic". On the right, it says "Last updated today at 10:15am".

**Summary**

In Progress	Medium
Status	Priority
10/18/2021	Yesterday
Date Opened	Date of Latest Action

**Evidence**

- Tests seemed to be irregular.
- Prior cancellations

**Recommended Actions**

- Offer transportation assistance (1 gap)
- Assign to social work (2 gaps)
- Nurse to call patient (2 gaps)

**Action History**

Status	Action	Date Assigned	Assigned by	Assigned to	Date Completed	Outcome
Schedule tests for the next 6 months		10:30 am	You	Broadway Health Scheduling		

Detail of patient records from HealthPointe's Digital Care product showing recommended actions that the interface can initiate. No real patient data was used to generate these screens. Proprietary. © HealthPointe Solutions. All Rights Reserved.

## 6 Integrating Principles in Real Life

Integrating these design principles into products and product development cycles can present an entirely different challenge. The culture of some work environments may not be supportive, particularly if they have achieved success with a technology-first approach. The principles described in this paper represent an ideal and unattainable state. However, championing even one or two of these ideas will increase the quality of the AI products we produce. Here are recommendations for navigating internal roadblocks to increase involvement and influence.

### 6.1 Start with Domain Expertise

Software and software quality engineers can champion Principles *Exhibit Confidence* and *Illuminate Logic*. Designers and product managers can explore *Facilitate Action*. Marketing can execute *Articulate Trustworthiness*, but the details about model behavior, data, data collection, data use, and established frameworks that algorithms incorporate should come from engineering.

Eventually everyone involved in the planning, design, and delivery of AI products can benefit from a shared understanding of all principles, particularly core principle: *Center Users*. Without the framework they provide, products can go off the rails quickly without any real awareness of the missed opportunities.

### 6.2 Choose Strategically

Target products where things are more flexible and open; where the roles of project team members are fluid, and contribution from multiple disciplines is welcome (Czernek 2018).

- Start with smaller projects rather than large ones. Quick wins can boost confidence and trust in the principles for use on future important products.
- Select from scratch products in the early planning stages or those that have newly formed teams. These can be easier to influence and introduce innovative ideas.

### 6.3 Plan Testing and Validation

Get involved in the planning process as early as possible. Consider users in project planning by asking the user definition questions from the *Center Users* section (5.1.1). If direct involvement is not possible, ask your manager to pose these questions. Write user stories correctly to facilitate easy testing.

### 6.4 Connect with Design

If your team has product or UX designers, it makes sense to build a relationship with them. It is mutually beneficial and will increase the quality of the products you deliver together. Ask designers if you can sit in on their early sessions and invite them to relevant engineering meetings. Maintain frequent connection to share status, ask/answer questions, discuss technical challenges, and co-create solutions.

## 7 Conclusion

There is increasing recognition that UX design principles have a critical role to play in the creation of AI products that delight users and make their lives easier while rebuilding trust (Agidi 2024). Along with an awareness that continuing to build technology-first products is unsustainable and ill advised.

Conversations about “Explainable AI” are gaining more traction and companies like IBM, Salesforce and OpenAI have added trust indicators and usage policies. However, interfaces demonstrating even one of the design principles described in this paper are rare.

There are so many opportunities, and it is the perfect time to advocate for a better way. The application of these design principles will usher in a new era of AI products. Pick a principle and start today.

## 8 Special Thanks

I would like to thank my colleague and friend [Nathan Gnanasambandam](#), Chief Technology Officer at HealthPointe Solutions, for his inspiration and support on this topic, as well as his friendship. We have worked together to build successful, user-centered AI products for over a decade.

## 9 Disclosure and Acknowledgement

This section discloses and acknowledges relevant third-party work from which the current work is derived; especially content generated with the use of AI and AI-assisted technologies.

Use of AI in the writing of this paper.

- No use of any AI or AI-assisted tools to create figures, images, or artwork.
- No images violate state or federal laws.
- Use of two AI or AI-assisted tools to help complete the research for this paper: Microsoft Copilot in the Bing browser and ChatGPT-4.
- Use of AI for specific use cases:
  - Editing proposal – After writing original, unformatted first and second drafts, ChatGPT-4 was used to organize and edit that content to fit the proposal format.
  - Improving paper clarity and impact – ChatGPT-4 was used to identify opportunities to reduce the use of the passive voice in the final draft.

## References

- Ajadi, R. (2024, April 13). *Designing for trust: a designer's guide to building responsible AI products*. Retrieved August 11, 2024, from Medium: <https://medium.com/@agidirukevwe/designing-for-trust-a-designers-guide-to-building-responsible-ai-products-9d5d0d07f85d>
- Chakravorti, B. (2024, May 03). *AI's Trust Problem: Twelve persistent risks of AI that are driving skepticism*. Retrieved August 11, 2024, from <https://hbr.org/2024/05/ais-trust-problem>
- Czernek, K. (2018, November 13). *Why Software Engineers Should Engage in Product Development*. Retrieved August 11, 2024, from Medium: <https://medium.com/hackernoon/why-software-engineers-should-engage-in-product-development-627d4d581795>
- Deodhar, S., Borokini, F., & Weber, B. (2024, August 5). *How Companies Can Take a Global Approach to AI Ethics*. Retrieved August 11, 2024, from Harvard Business Review: <https://hbr.org/2024/08/how-companies-can-take-a-global-approach-to-ai-ethics>
- Donachie, P. (2024, March 18). *SEC Accuses Two Firms of Lying About AI*. Retrieved August 11, 2024, from WealthManagement.com: <https://www.wealthmanagement.com/regulation-compliance/sec-accuses-two-firms-lying-about-ai>
- Drapkin, A. (2024, August 16). *AI Gone Wrong: An Updated List of AI Errors, Mistakes and Failures*. Retrieved August 20, 2024, from tech.co: <https://tech.co/news/list-ai-failures-mistakes-errors>
- Gartner Research. (2024, June 17). *Hype Cycle for Artificial Intelligence, 2024*. Retrieved August 15, 2024, from Gartner: <https://www.gartner.com/en/documents/5505695>
- GeeksforGeeks. (2024, August 8). *What is Artificial Intelligence – AI Definition, Types, Working - GeeksforGeeks*. Retrieved August 11, 2024, from GeeksforGeeks: <https://www.geeksforgeeks.org/artificial-intelligence-an-introduction/>
- Heikkila, M. (2024, July 22). *AI companies promised to self-regulate one year ago. What's changed?* Retrieved August 11, 2024, from MIT Technology Review: <https://www.technologyreview.com/2024/07/22/1095193/ai-companies-promised-the-white-house-to-self-regulate-one-year-ago-whats-changed/>
- Majewski, S. (2022, August 16). *Explainable AI: A Way to Explain How Your AI Model Works*. Retrieved August 18, 2024, from Medium: <https://medium.com/dlabs-ai/explainable-ai-a-way-to-explain-how-your-ai-model-works-5766b1f11deb>
- Pratt, Louron. (2024, July 17). *Product survey findings: Only 15% of users are embracing AI features*. Retrieved August 20, 2024, from Mind the Product: <https://www.mindtheproduct.com/product-survey-findings-only-15-of-users-are-embracing-ai-features/>
- Timbo, R. (2023, September 28). *AI Bias: What It Is & How to Prevent It?* Retrieved August 11, 2024, from Revelo: <https://www.revelo.com/blog/ai-bias>
- Tonye, G. (2021, August 25). *Machine Learning Confidence Scores — All You Need to Know as a Conversation Designer*. Retrieved August 11, 2024, from Medium: <https://medium.com/voice-tech-global/machine-learning-confidence-scores-all-you-need-to-know-as-a-conversation-designer-8babd39caae7>
- W3C Web Accessibility Initiative. (2024, August 6). *How to Meet WCAG (Quick Reference)*. Retrieved August 11, 2024, from W3C Web Accessibility Initiative: <https://www.w3.org/WAI/WCAG22/quickref/>

# Scaling Test Automation using a Metadata-centric Architecture

**Donald Maffly**

dmaffly@hotmail.com

## Abstract

When systems are initially designed, scalability is a consideration that is often overlooked. The first priority is simply to deliver something that meets customer requirements, and everything else is secondary. As a system begins to grow and evolve over time, however, it can become increasingly difficult to meet customer requirements because the system is not able to scale effectively. Lack of scalability creates a bottleneck presenting different obstacles for different functions within a product team (BSAs, Developers, QA, Production Services). For QA it can mean that automated tests become increasingly difficult to maintain, more challenging to trouble shoot, more unruly to navigate, and too time consuming to run.

Refactoring a system to be metadata-centric can effectively address this bottleneck and transform a system into one that is truly scalable. Metadata is data that describes various system components; QA can leverage this information to write generic, metadata-driven automated tests. Consequently, lines of test code are drastically reduced, and ironically, code coverage can actually be dramatically increased. Test code becomes much easier to navigate, to maintain, and to run.

## Biography

Donald Maffly works out of Portland Oregon as a QA consultant. He brings over 3 decades of experience as a software developer, test developer and manager. Recent areas of focus have been in business intelligence and data analytics serving both the financial and healthcare industries.

*Copyright Donald Maffly 2024*

## 1 Introduction

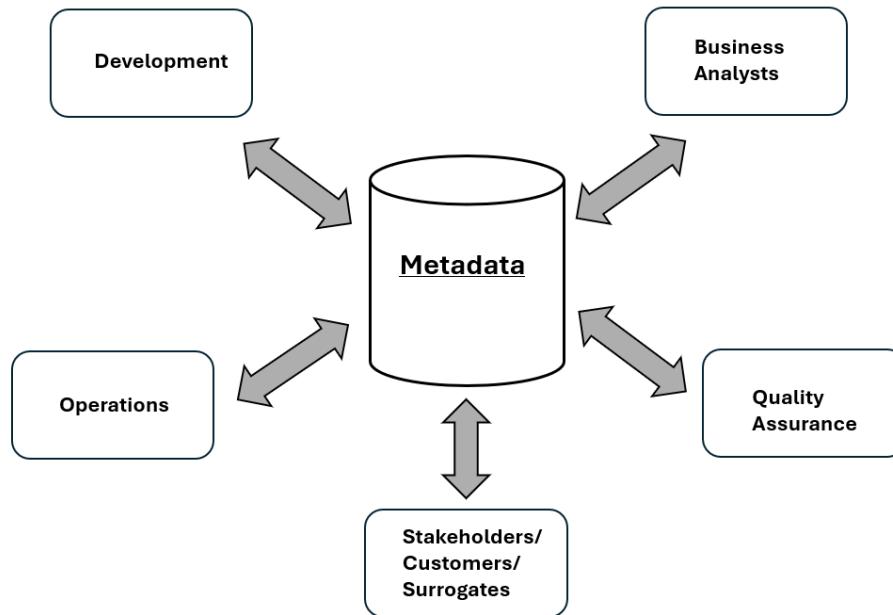
When systems are initially architected, a sometimes narrow set of customer facing requirements can be favored over other internally originating requirements. More often than not, it is likely speed-to-market that trumps all other concerns such as performance, security and scalability to name just a few. As a system grows, however, it can be more difficult to extend and maintain it because initial design was not done with extensibility in mind. This shortcoming rears its head in the form of longer develop times, longer test times, longer operational times and quality may even suffer.

As you dig into code, you might also observe the same old patterns repeat themselves; you might find highly detailed data (e.g. directory paths, file names, etc.) hardcoded into programs. Lines of code can multiply several times over, and with it, code becomes harder to navigate. It may take longer and longer for a system to run through its computations. These are just a few of the telltale signs that a scalability bottleneck is looming.

Part of this is just natural system evolution. Mistakes may have been made earlier that need to be corrected, or perhaps system requirements changed dramatically over the course of time. The team refactors, and the system evolves. That said, regardless of whether it is during upfront system design, or refactoring code years later, sound software engineering principles apply; sound high level architecture, modularized and divided into domain specific abstractions. Common code can be identified and from there consolidated and condensed into a single abstraction. Instead of having thousands of lines of code reinventing the same old wheel, you now have a single piece of code that performs the same task. Lines of code become drastically reduced, the code is easier to understand and to navigate, and ultimately to develop, test and operate.

This is where *metadata* comes in. One way to understand the concept of metadata is as an abstraction mechanism, and as such it is a cornerstone of sound software engineering. If metadata is given the proper visibility, it can become the hub around which an entire organization can anchor itself when undertaking an ambitious engineering endeavor (see figure 1 below). In this paper, we define metadata, explain how to create it, and provide an example illustrating how metadata can be leveraged to benefit QA and the testing effort.

*Figure 1.*



## 2 Metadata : What is It?

Metadata is hardly the latest new big thing. The *meta* concept has been around for quite a while in computer science circles. **Meta classes** are used in the Smalltalk object-oriented language to create classes of classes to control the computational model of Smalltalk. **Meta languages** (a.k.a. formal grammars) can be used to generate language parsers, yacc is good example; it takes a formal grammar as input and generates a parser from it. **Meta programming** is writing programs that generate other programs; these can be as simple as macros, Java generics, C++ templates, or dynamic SQL, but could also be sophisticated AI programs written in Prolog or LISP that are generating programs or even extending themselves. And then there is **metadata**, the best known example might be in a RDB (relational database) context where a data schema (describing table structure, columns, relationships) is captured itself in the form of data (i.e. tables, columns, etc.).

The Oxford English language dictionary defines ‘meta’ as ‘referring to itself or to the conventions of its genre; self-referential’. So strictly speaking metadata is data that describes data. In the context of this paper, by metadata, we mean data that describes the system under test (SUT) *itself* and furthermore, data that describes *tests* targeting the SUT.

Metadata about the SUT *itself* might contain:

- descriptions of system entities (e.g. files, directories, processes)
- relationships between entities
- higher level abstractions that add deeper hierarchical understanding; e.g. classifying entities into logical groupings
- input descriptions
- output descriptions
- system change history

Metadata about *testing* the SUT might contain:

- descriptions of testing entities (e.g. validation scripts, test results files & directories)
- relationships between testing entities and SUT entities
- higher level abstractions that categorize testing entities into logical groupings

Consumers of metadata range anywhere from development engineers (charged with architecting and building the SUT), QA engineers (charged with testing the SUT), Production Services engineers (charged with running the SUT in a production setting), to business analysts (charged with representing end users of the system and coming up with system requirements).

Metadata can be contained in a relational database, but it doesn’t necessarily need to be. It could also be contained in structures within a program, or it could be contained in raw text files. Regardless of how and where metadata is stored, what is critical is that the metadata is easily accessible in some form – preferably via a programming API. Once accessed, metadata can be used to:

- drive test automation of the SUT
- inform the development of the SUT
- drive the running of the system in development/test/production environments
- provide understanding of the inner workings of the SUT
- provide a historical record of changes made to the SUT

## 3 Metadata: Creation

Early in a system’s life cycle, it can sometimes be hard to identify metadata, but generally a good place to start, is using the system architecture itself as a guideline. Which system abstractions are central, and ones that you want to publicize to stakeholders, developers, testers, and operators alike? These might be:

Excerpt from PNSQC Proceedings

Copies may not be made or distributed for commercial use

PNSQC.ORG

Page 3

be particular processes, input files, or output files. But sometimes it may not be until a system has grown and evolved that metadata opportunities can become identifiable. Only then, may recurring patterns emerge that are suitable to express as metadata.

Recall that one of the benefits of using metadata is to capture data describing the system in a single place, so that you don't have to sift through thousands of lines of code to understand how the system works. Not only can metadata aid in understanding and navigating the system, but it makes modifying the system much easier. When changes can be confined to metadata, often times system changes can be done quickly, and perhaps even dynamically during the operation of the system, not requiring any development or QA effort best case.

To this end, we want to capture metadata information that expresses small differences in different commonly occurring classification of objects in the system that you want to give broader visibility to. These object classifications might be:

- directories where input/output files live
- the structure of filenames
- how the file is used
- the relationship between objects in the system
- when processes are run
- the order that processes are run

Armed with metadata situated in a single place, developers are able to write reusable code, testers to write generic test cases, and system operators are able to drive the running of the system.

## 4 Metadata: an Example

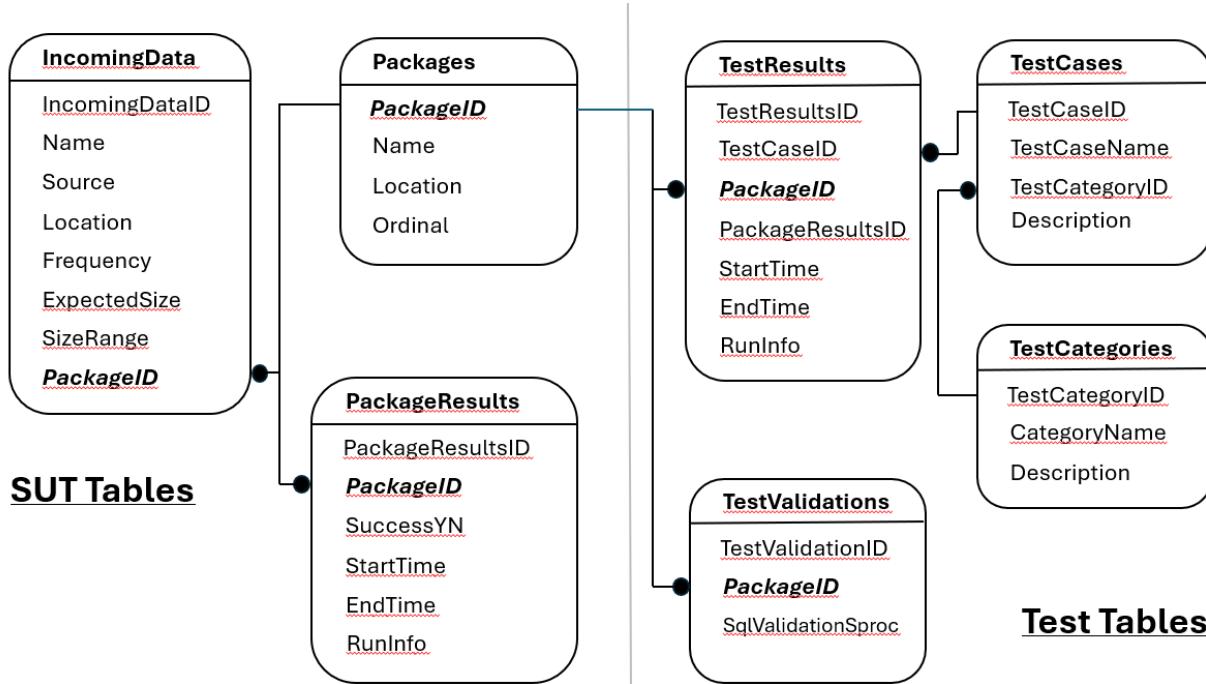
A large regional bank in the Pacific Northwest has a data warehouse system built on the Microsoft stack (SqlServer, SSIS, Azure Dev/Ops). It ingests data from many cooperating financial institutions, and processes it via ETL (Extract, Transform, Load) processes where data goes through several refinement phases. Finally, data lands in a data warehouse, where it can then be served via data visualization tools like Power BI to bank executives to help them make informed business decisions.

This data ingestion process is complex and must be done within a tight time frame bounded on one side by the time the last incoming file arrives and on the other side by the time consumers in the bank need to access the processed data in the data warehouse. Other complexities include

- ordering dependencies exist between ETL processes
- frequency that incoming files/data are delivered vary
- incoming data is delivered in a variety of different formats (flat file, Excel, other databases)
- incoming data format of incoming data changes occasionally
- incoming data from a particular source can straddle multiple files

The entity relationship (ER) diagram (see figure 2) depicts metadata tables which are designed with these requirements in mind. These tables capture essential SUT information that varies between data ingestion processes and as such can be fed in as parameters to generically written code, be it implementation code, test code, or operational code. Furthermore, it enables the system to be easily configurable. Thus, changes do not necessarily need to require a major effort on the part of either development or QA.

Figure 2.



In the ER diagram, observe that SUT tables are organized on the left side, whereas test metadata tables on the right; this separation is intentional, so that test tables can be deployed separately, as they are not something that belong on a production system. Furthermore, dependencies between SUT and Test tables are uni-directional, i.e. Test tables depend on SUT tables, SUT tables do NOT depend on Test tables.

Observe that the *Packages* table is the essential target of the testing effort as well as the overall running of the system, and as such it is centrally located in the ER diagram; foreign key relationships (*PackageID* highlighted in bold italic font) to the *Packages* table exist in four other tables—both SUT tables and test tables.

SUT metadata tables contain metadata that is specific to the SUT; the *Packages* table (i.e. representing SSIS packages) describes peculiarities with the packages such as their name, location and ordering dependencies. The *IncomingData* table describes the different types of data sources that are associated with each package. And the *PackageResults* table are where package results are logged.

Similarly, test metadata tables contain data which is used to inform the test effort and ultimately to drive the running of tests. Table *TestValidations* maps SQL validation scripts (as stored procedures in the database) to be run against particular packages; *TestResults* contains pass/fail results. Higher level testing abstractions are contained in tables *TestCategories* and *TestCases*, which are used to drive the running of the tests.

For the sake of this paper, we have abridged the metadata diagram, so that we could more easily convey the benefits that metadata brings. In actual practice, however, metadata could be more complex. For example, the banks datawarehouse system outputs data in different formats, and we have excluded that here. Also, as far as test metadata is concerned, there are likely to be packages or classes of packages to which a certain test case can be run on—in which case a new package group abstraction (and table) could be added, which would group packages together, and a package grouping foreign key would be added to the *TestCases* table. This also has been omitted from our ER diagram for the sake of simplicity.

Together both SUT and test metadata above create the foundation for which generic test code can be written (in pseudo code) below. The example below demonstrates the coding of a happy path test case to be run against all packages which have run frequency = "Daily". This test is written to run all the "daily" packages, but could be reworked to take a *PackageID*, or even a package grouping id, and run a smaller subset of packages:

```
Procedure TestRunPackagesDailyHappyPath(<params>)
{
    Get all SsisPkgs with Frequency='Daily'
    Order these packages per Ordinal value;
    For each pkg in this ordered list of packages
    {
        From IncomingData get input filename and drop directory, and drop it
        Run the package;
        Check for run status in PackageResults;
        Run any validation scripts associated with the pkg;
        Log PASS/FAIL test status in TestResults table
    }
}
```

Here is a failure test case coded to verify that a package behaves as expected (i.e. fails) when an expected incoming file is *not* found:

```
Procedure TestRunPackagesMissingFile(<params>)
{
    Get all SsisPackages with Frequency='Daily';
    Order package in SsisPackages table per Ordinal column;
    For each pkg in SsisPackages
    {
        From IncomingData get drop directory, and remove any files there
        Run the pkg;
        Expect run status error;
        Log PASS/FAIL test status in TestResults table
    }
}
```

Other failure tests that could be written could also be metadata driven. For example, incoming files could be dropped that don't conform to *ExpectedSize* and *SizeRange* parameters; using the *Ordinal* column, packages could be run in the wrong order. When SUT failures are not encountered during run, this would constitute a test failure.

The big takeaway here is that the tester is relieved from coding separate test cases for each SSIS package, thus reducing code bloat by possibly as much as an order of magnitude. Test code becomes much easier to maintain – particularly when SUT changes are restricted to just the *Packages* and/or *IncomingData* tables. Also, other gains are made by having system metadata and testing metadata co-located. The most obvious of these is during test failure analysis (which requires human intervention), using the *TestResults* table's direct references to *PackageResults*, we quickly get one step closer to determining root cause.. Furthermore, co-location of system and testing metadata fits hand in glove with agile development practices, where test driven development and pairs programming are the norm. A synergy between development and testing is enabled and productivity gains are to be had.

## 5 Metadata: other benefits to the testing effort

### 5.1 Gauging Test Coverage

Test coverage metrics provide a critical SDLC function, and act a measuring stick for the completeness of a test effort, and ultimately for the quality of the SUT itself. Calculating test coverage metrics can be challenging particularly for large, complex systems, and particularly when there do not exist direct links between test cases, test code, and system source code. Traceability from test case to SUT function becomes muddled.

Manually scripted test cases can provide traceability by *name* to a functionality or a given scenario. Also, links between test cases and agile ‘stories/epics’ provided by collaborative development environments such as Atlassian and Azure DevOps are genuinely very useful. But naming references and links are often too distant from the actual system itself to be entirely credible. Furthermore test results can be output to an entirely different media and need to be parsed and then painstakingly mapped back to SUT functionality to really understand how test results relate to the SUT as a whole.

By contrast, metadata can be used to make more direct, and hence more credible, links between an actual process function in the SUT and actual tests, since the same metadata properties are used to drive both, and they can be documented in a test results table with direct references to metadata entities. If such a test results table lives in a database, then a simple SQL JOIN query between test results and SUT metadata make it very easy to demonstrate what parts of the SUT were tested as well as which were not.

### 5.2 Generating Test Plans from Change Sets

With large systems it can be simply too time consuming and ultimately too costly to run entire regression test suites (along with the time to analyze test failures) to be practical. If code changes to a system are narrow in scope for a given production release, then it is reasonable to assume that so too could a corresponding test plan be made narrower, and thus less time consuming and costly to run. For example, revisiting the bank data warehouse system example, suppose a change were made to a single financial institution ingestion module, if metadata existed that mapped system dependences to test cases were in place, a suite of test cases (from the TestCase table) could be computed. Such dependencies could be at a module level, or even down to the smaller granularity of implementation files (e.g. C#, SQL, etc).

### 5.3 Generating Test Code using Generative AI

Generative AI tools like ChatGPT have demonstrated that given problem description, it is capable of generating small programs in a wide variety of programming languages. There is no reason to doubt that we are close to the day when generative AI could generate programmed test cases, if that day is not already upon us.

The key to making that work is providing generative AI quality information as input that it needs to code a test case. Metadata fits that bill very nicely; it provides structured data that is easily digestible. Metadata coupled with human language guidance should very easily provide generative AI the direction it needs to generate relevant and useful test case code.

## 6 Concluding Remarks

At our regional bank, we observed the following before vs. after improvements after QA embraced the metadata concept:

- lines of test automation code were reduced by at least an order of magnitude and hence much less time was spent maintaining the code

Excerpt from PNSQC Proceedings

Copies may not be made or distributed for commercial use

- test pipelines experienced many fewer false fails, and with that time was saved not going down rabbit holes tracking down root causes
- automation code was much more robust and could easily sustain changes to the underlying SUT without breaking—also a big time saver

While this paper has focused mostly on QA's use of metadata to supercharge the testing effort, it goes without saying that QA does not occur in a vacuum. QA is intimately connected with the rest of an organization; requirements/acceptance criteria authored by BAs inform QA's test case writing, operations and QA share the similar concern of simply running the system, and developers and QA might even share the same cubicle in paired programming agile development situations.

If the language of system metadata becomes the *lingua franca* of these interacting communities, then there becomes a lot less room for misunderstanding, and a lot more opportunity for collaborative and highly productive work. And finally, bringing generative AI tools into this mix has the potential to supercharge this collaboration and the engineering effort as a whole.

## References

- Goldberg, Adele. 1983. *Smalltalk 80: The Interactive Programming Environment*. Boston: Addison-Wesley Publishing Company (now Pearson).
- Levine, John R, Tony Mason, and Doug Brown. 1992. *Lex & yacc*. Sebastopol, CA: O'Reilly and Associates.
- Stroustrup, Bjarne. 2013. *The C++ Programming Language*, Sebastopol, CA: Addison-Wesley Professional (O'Reilly).
- Abrahams, David. 2004. *The C++ Template Metaprogramming*, Sebastopol, CA: Addison-Wesley Professional (O'Reilly)

# Bringing the Dawn of Security & Compliance to Project Teams

**Vivek Sahai Mathur**

mathurvs@outlook.com

- **Abstract**

The majority of outsourced work is piecemeal: phases of the SDLC are carved out and Service Provider companies deliver the output to the customer as per the Statement of Work. The Customer then integrates all the deliverables together. The Service Provider's focus is on speed, and ensuring that the business requirements are covered. This approach leaves out contract compliance and data protection, Different nations around the world are adopting data security and privacy regulations. Violations are penalized, so customers are transferring the risk through stricter enforcement of contractual penalty clauses.

While the legal framework and insurance is being put in place by the Service Provider's corporate teams, trying to "Inspect in" or "Audit in" contractual compliance is time consuming and ineffective for the project delivery teams. Typically project management does not appreciate the root causes and impact of the risk. Process quality and delivery quality teams also do not cover these aspects a priori.

This paper walks through my experience in creating, piloting, and rolling out an org-wide initiative to awaken the PM community to non-functional requirements and customer's implicit security and privacy expectations. A lightweight framework for early detection of gaps, tracking of corrective actions, and feedback on residual risk for the project was designed to be easy to administer, and quick for the PM to respond, with most information available in the project definition.

- **Biography**

*Vivek S Mathur started his IT career in 1997. With 30+ years of varied work experience ranging from the Government to software product and service companies, his focus has always been on setting up robust processes, and seeing their impact on the quality of deliverables and productivity. Making changes that delivery teams internalize and adopt, and add value to the organization.*

*Vivek has set up core software development processes for product and service companies like McAfee, Intelligroup & Altisource Labs, and has set up and led large delivery teams for multiple customers.*

*Vivek has always tried to give back to the community, and has been associated with STeP-IN Forum ([www.stepinforum.org](http://www.stepinforum.org)) actively for the past 11 years. He is currently the President, STeP-IN Forum*

*Experienced in Information Security and Software Quality, and in leveraging these to achieve highest productivity and predictable project delivery while leveraging the partner ecosystem.*

Copyright

 Vivek Mathur

15 Jun, 2024

## 1 Introduction: The Blinkered PM

The typical IT outsourcing deal pursuit crosses multiple stages and has many handoffs. Teams with different responsibilities review the documentation and participate in the negotiation rounds. As the proposal process winds towards its conclusion, multiple aspects of the deal are accepted by these teams.

The result is a signed Statement of Work (SOW), and a project team is assigned to deliver the software application to the customer. When the project is ready to begin, the chosen Project Manager builds the team. At this point in time, the PM is aware of:

- Technology in use (and therefore skills to include in the team)
- Estimated effort and proposed schedule (therefore team size and onboarding duration available)
- Location of the documents (covering requirements and architecture)

And, in certain more mature organizations:

- Expected project contribution margin (and profitability targets for the Business Unit and company as a whole)

As the PM works to get the project started, the focus is on the immediate needs, while other aspects of the deal that have been accepted are not communicated, nor prioritized. The PM begins the project with incomplete appreciation of these expectations and commitments. Primary among these are:

- Information Security and Data Privacy
- Regulatory and Compliance commitments
- Non-functional requirements

The focus of this paper and the assessment it explains, is on the first two topics. Non-functional requirements need to be elicited as part of the project discovery stage, and taken care of as part of the deliverables.

## 2 The Background Scramble

As mentioned briefly above, Information Security and Data Privacy are implicit customer expectations and an unidentified risk for complete project delivery. These two aspects have taken on greater importance in the current global environment.

### 2.1 Elevated Cyber Security Risks

#### 2.1.1 Setting the Context 1: The Changed Attack Landscape:

In the current cyber environment, cyber defense needs and the scope of control mechanisms are very fractured, while at the same time requiring continuous attention throughout the SDLC. In short:

- Perimeter Defense is no longer adequate
- 'Fit and Forget' defense is no longer adequate
- Retrofit after Deployment is no longer adequate
- All environments are open to breach

To counteract these risks, we need to adopt a Defensive Security stance that:

- Aligns security to business requirements

- Designs for minimal attack surface
- Builds in security
- Deploys for defense in depth

The earlier we start to Build in Security and inculcate a defensive mindset in the entire team, the less risk will remain at the time of project delivery. We can incorporate the design, implementation, and review effort into the proposed solution, or raise these aspects during initial ideation discussions with the customer.

### **2.1.2 Setting the Context 2: The Competitive Landscape:**

In a competitive industry, customers are looking for a partner to advise, guide and keep their best interests front and center. The best partner will be one to use historic and concurrent experience and industry knowledge, leveraging exposure to multiple industries and cross-leveraging to help identify new solutions and industry leading best practices.

Since Cybersecurity is a growing risk in an increasingly mobile and connected world, applying the ability to proactively reduce cyber disasters for the customers will increase credibility as a trustworthy and worthwhile partner.

## **2.2 Forced Risk Acceptance**

The Service Provider must balance the revenue imperative with the risk of a security incident, and as such is faced with a dilemma. While risk transfer through the use of insurance policies is available, it is an expensive solution, and the customer specifications for insurance terms and conditions make it more complex with every passing year. Inevitably, risk is accepted, and the pressure is on the delivery team to implement the risk mitigation to the maximum extent possible. However, without prior information and understanding of the risks that have been accepted by the company, it is very difficult for the project team to come up with any cogent and useful approach to manage and control the risks.

Therefore, it is imperative to unearth expose the risks at the earliest. For this purpose, we created a risk assessment framework.

# **3 The Risk Assessment**

## **3.1 Background**

The risk assessment application was based on a request from Senior Management to create a corporate security and compliance risk landscape for all in-flight and upcoming projects. During the design stage, the following decisions were taken:

- a) Design an end-to-end framework for proactive project risk assessment:
  - a. Keep the process simple enough to be quick, but provide indicators to all or most potential problems.
  - b. Integrate with preexisting ticketing workflow and management reporting process
  - c. Provide a visual aid to the assessees to track progress. This shows how much controllable risk remains, and keeps the team motivated to complete the assigned interventions.
  - d. Add the risk assessment as a part of the project initiation process.
  - e. Structure the intervention as a self-assessment for scalability, using simple language and concepts.
  - f. Definitely NOT to follow the ISO certification audit approach. The initial clauses of the standard focus on organizational mandates and structures, which are beyond the scope and responsibility of the PM, who is busy in the daily project delivery tasks.
- b) Monitor the desired outcomes
  - a. Expose hidden improvement opportunities in project delivery related to information and data security risks

- b. Educate the project leadership to contractual and regulatory aspects of their work, and how cyber security risk that the project team adds to the deliverables can have a much larger impact on the company. Risk may not be limited to the next billing amount, but could have real financial consequences if a penalty clause is triggered.

The Assessment Model was designed to add value to multiple stakeholders:

- Technical team members: Implementable & clear, prescriptive directions
- Project Managers: Delivery interventions during the project timeline
- Portfolio Managers: Risk Governance and Corrective recommendations
- CXO level: Corporate Strategy inputs and portfolio risk management

To try and address this spread of detail and abstraction, and differing focus areas and objectives, the following options were evaluated to achieve the quickest start and roll out.

## **3.2 Options evaluated:**

### **3.2.1 Leveraging Standard Security Audits:**

Existing models for Security Assessment such as ISO27k, etc. are designed with an Outside-in perspective, and are useful for 2 sets of users:

- Internal Auditors looking at their internal software development processes
- Third party auditors identifying lacunae in software development process and abilities

Security Audits are fundamentally “After the Fact”. This is a result of the implicit expectations for the audit, and part of the “Nature of the Beast”:

- a) ISMS has to be defined and implemented
- b) Projects should already be following the ISMS recommendations, or in process of adoption.  
In most cases, conformance with the ISMS is not complete, and is always in a state of flux for the Service Provider.
- c) Defining, Deploying, and Monitoring the ISMS is typically the responsibility of a specialized Task Force.
- d) All project teams cannot participate in the task force
- e) Projects end and knowledge is lost
- f) New projects start constantly
- g) New projects often follow the customer’s processes.

Security Audits also define desired states, but do not recommend actions to be taken by the project team. An audit is typically driven across the company, and focus on individual projects and project teams is limited to closure of non-compliance reports (NCRs). Ensuring adequate technical depth and rigor of the solutions becomes secondary to meeting the audit schedule.

### **3.2.2 Utilizing Standard Risk Frameworks (e.g. OWASP/ SAMM/ CIS):**

They are mostly derivative from the ISO standards in terms of the areas covered.

- a) As in the case of the external audit-based approach, the frameworks do not have a practitioners viewpoint to guide how to avoid security issues or build a secure product.
- b) Cannot be used to assess new projects, but need some deliverables for review.
- c) Structured as maturity models, and yield a score that talks about the capability of the project to deliver secure output, rather than interventions that are needed to improve deliverables.
- d) Do not address the project delivery process at all. The standard framework was modified with delivery related aspects such as Training & Education, Organization & Reporting, and Process implementation.

### **3.2.3 Custom-built model:**

The Service Providers need is to ensure that existing and future projects and work products are made compliant with security best practices. For this purpose, a custom-made framework works best to provide project teams with a simple, quick assessment mechanism, a convenient method to submit their results, and an automated data ingress and analysis mechanism.

- Risk areas are limited to those under the project manager's control
- Project delivery aspects are covered
- Information about the risk areas is available as part of the project initiation process and the assessment questions can be answered by the PM and project architect within the first 20 days of the project start date.
- The solution also provides senior management with a comprehensive dashboard to show overall metrics and provide click-through to project level details and the ability to slice and dice across multiple dimensions.

## **3.3 Criteria for Evaluation**

The assessment model was assessed for ease of implementation and adoption, using the following evaluation criteria:

### **3.3.1 Effective**

- Identifies maximum risks in one pass.
- Forces evaluation of individual risks
- Triggers an analysis of the risks caused by interaction of different aspects of the project characteristics.

### **3.3.2 Easy to incorporate in the project life cycle**

- Aligned with and synergistic to existing project management and portfolio governance processes
- Intervention Triggers clearly defined and monitored externally
- Learnings can be easily transferred to the next project(s)

### **3.3.3 Clear and prescriptive**

- Understood by tech and non-tech project roles
- Clear actionable Recommendations applicable at project level
- Inculcating risk awareness and risk management focus in the team
- Discussion based; all doubts cleared during the assessment

### **3.3.4 Simple to administer**

- Self-explanatory
- Completed in ½ to 1 hour
- Easy to track
- Clear visualization of results and progress

### **3.3.5 Comprehensive**

- Technology agnostic
- SLDC agnostic
- Delivery type agnostic: Design, Dev, test, deploy, maintain/ support
- Covering code and 'Beyond Code' aspects of project delivery (Not just the SDLC)
- Covering project and portfolio governance aspects: Corporate context means that projects impact not only the immediate deliverable but also the customer relationship, market reputation, and ability to acquire new leads once the client contacts move to other corporates.

- Providing opportunities to advise the client: Showcase thought leadership and cross leverage learnings, with business domain focus
- Robust & Industry Leading
  - Aligned to Global Best Practices
  - Capable of evolving and staying relevant

### **3.4 Self-Assessment Design & Implementation**

The application was implemented using standard collaboration and analytics tools: Microsoft Teams for meetings, and PowerBI for reporting. An Excel spreadsheet was created for data entry, sent as an email attachment. The combination of tools provides an easy way to conduct the assessment with geographically distributed teams who may not have access to the corporate network.

#### **3.4.1 Core Excel App**

##### **a.) Assessment Entry sheet**

14 factors in 3 areas were identified as being typically in the blind spot of the PM when evaluating Information security risk. An indicative response spectrum was defined for each as a guideline for rating selection.

###### **1. Technology related:**

- a. SDLC span: stages of the SDLC covered
- b. Tech Stack depth: business logic to server management
- c. Non-production environment control: the extent to which management of code promotion was in the project's control
- d. SDLC model: time available to fix findings
- e. COTS integration: the extent to which the business logic was self-coded.
- f. Security Requirements: Whether customer has explicitly noted down security aspects to be addressed.

###### **2. Demographic factors:**

- a. Industry Sector: Inherent complexity and focus on security
- b. Regulatory Requirements: Compliance to regulations and standards
- c. Data Privacy impact: applicability of data privacy regulations from various countries
- d. Data Components: Personal and non-public information being processed in the application
- e. Usage/ Impact: extent of control on the user base behavior

###### **3. Contractual Factors:**

- a. Deliverable responsibility: The extent of accountability for outcome vs effort
- b. Project duration/ iterations: availability of time to fix issues
- c. Direct Financial Impact: Penalty amounts and clauses

Each factor is assessed by the project team along with the representative from the delivery quality team. In the initial stages, the author would participate in person, along with his team, later the assessment was conducted by the project team and delivery quality teams.

Scores were assigned from a non-linear scale to assist in clearer prioritization. Based on the score, the author mapped the factor scores to the relevant security controls.

Focal areas for corrective action were identified for each factor and rating

#### **b.) Typical values**

To assist the PM and project architect choose the correct score, two guidelines were provided:

- Typical values for each factor, from least to most risk, giving the team a mapping to typical project situations and SOW characteristics.
- A strong suggestion to pick the more severe rating value in case of any disagreement.

#### **c.) Map to Action items**

Each rating value was mapped to actionable interventions that were in the control of the project manager and team. The assessment was conducted within the first 20 days of project initiation and for projects with an expected duration of more than 3 months to give the team sufficient time to implement the changes and realize benefits.

- 2 globally accepted models were used:
  - CIS: The Center for Internet Security (CIS) is a nonprofit organization with members including large corporations, government agencies, and academic institutions. Initially nurtured by NIST and OWASP, CIS is now independently managed to provide a free and fair assessment and responsiveness to changing cyber landscape.
  - BSIMM: Industry benchmark body monitoring the actual cyber-security actions undertaken by the member companies. BSIMM controls are constantly being reviewed and mapped to other cyber security standards
- In addition, to provide a compilation and easy reference to standards and controls, we included a reference site from the University of California, Irvine, which has been made Open Source.

The Action items identified were raised as tickets in the Delivery quality Ticketing Tool (initially in-house and later using a third-Party Software), and subject to the same severity classification, closure timelines and escalation process as all other tickets, ensuring that the security assessment received the same attention as any other ticket raised.

#### **d.) Integration to Executive Dashboard**

Data from each assessment was added to the database and used as input to the Executive Dashboard portraying the Security Risk Landscape for the company. Accumulation was at different levels of abstraction, from project code, Service line, Business Unit and up to the corporate security risk profile. The dashboard highlighted the common areas of concern, like secure development lifecycle, data privacy, and contractual commitments.

#### **e.) Action tracker**

To avoid surprises, the assessment repeats every quarter, and on any change in scope.

To make the process usable and ensure the team's engagement on an ongoing basis, especially for long running "steady-state" projects where there is little or no change in scope, technology or vendor responsibility, a visual aid was created in Microsoft Excel to track the progress and showcase how close the team is to reaching the end state of "no more interventions available." This tracker gets the inputs from the database and is refreshed after every assessment meeting.

## 3.5 Building Capability & Capacity for Rollout

The implementation of the assessment across all new projects has two challenges: Capability and Capacity

### 3.5.1 Imparting the skills to review the assessments

The Delivery Quality team was chosen as the ideal team to participate in the assessment because of their frequent interactions with the project teams throughout the project duration. Most of these interactions take the form of data gathering and information capture, into standardized formats for audit and reporting purposes. The Risk assessment falls seamlessly into the same category, albeit with a different focus and requirement of rating evaluation.

Delivery quality team also has follow up meetings to close on the Action Items that were raised in the project ticket tracker.

A training session was conducted by me to sensitize the team to the following factors:

- How not to accept exemption requests from PMs: If there is data being used, and algorithms being applied, some form of SDLC will be in use, and Information Security Risk or Data Privacy considerations will be applicable
- Trends in code development and application deployment environments, and the impact on the exposed risk surface. (This evolution was covered in my paper from PNSQC 2023: [https://pnsqc.org/vivek\\_mathur\\_2023.php](https://pnsqc.org/vivek_mathur_2023.php))
- To insist on risk identification: Threat Modeling approach using STRIDE
- The 14 factors and their inter-relationship. How the factors cover almost all aspects of software delivery that the PM and project team can understand and control.
- Talk through the sample ratings and help the team relate to their project reality
- Show the relevance of the factor -> action items mapping
- How not to accept the self-ratings by the project teams, how to be skeptical and get a more realistic rating.
- Technical debt will persist, leading to repeated high risk ratings, and a higher residual risk score.

### 3.5.2 Ensuring capacity to cover all new projects

Monthly, the number of new projects to be covered by the assessments was substantial. With the assistance of delivery quality team, and my dedicated handful, we were able to address all fresh starts where the assessment was found to be applicable. Renewals, short term, small and POC projects were exempt.

Each assessment took less than an hour, and meetings were held after the information pack (assessment worksheet, instructions, and FAQs) was sent to the PM one week in advance, and both PM and architect were available to participate in the meeting with the self-ratings filled in.

## 3.6 Process & Governance

The steps in the process towards a born secure Company are:

- Company Project Portfolio Survey - Identify existing projects and New Starts
- Risk Identification & Quantification - Focus on the relevant factors for each project. This is the core of the process and requires active participation from the project management and technical teams.
- Apply security controls - Risk based prescribed controls have to be implemented during the course of the project to fix weak areas and enhance security stance
- Monitor and repeat - every project milestone or half yearly
- Review and Improve - Conduct retrospective meetings and feed back the learnings into the process for continuous improvement

### **3.6.1 Risk Identification & Quantification Meeting:**

We need to identify the influencing factors for the project. The scope and depth of security activities will depend upon the characteristics of the project, or based on the information provided in the RFP documents.

### **3.6.2 Control Implementation Status Meetings:**

Once the relevant controls are identified and being implemented, the residual risk is assessed periodically at project milestones and quarterly/ half-yearly. Completion of the controls implementation is tracked and status is updated to the Teams workspace for progress tracking, analysis and reporting. One assessment just before final delivery of the project ensures that all possible actions have been completed, and the attack surface is minimized.

The assessment values are appended to the database at the granularity of Project – Date – type of assessment (Initial, Ongoing, Pre-delivery), and latest status is reflected in the Excel Action Tracker.

### **3.6.3 Process Review Meetings:**

For the purposes of process improvement, a retrospective meeting is used to capture the final project status and the "Stop/ Start/ Continue" analysis of the process itself, including updates to the List of values in the dropdowns, the file handling, and the reporting improvements.

## **4 The Wide-Angle View**

### **4.1 The Payoff to the PM**

- The primary payoff to the PM is a better, more secure deliverable, with lowered chances of a in-Production firefight to add a security Band-aid.
- Secondary payoff (longer term) is better credibility as a security-oriented delivery personnel.
- Being able to think in terms of risk makes the PM much more likely to have a seat at the table for architectural discussions in the future., having a say in the future direction of the application roadmap.
- Reduced technical debt in the product backlog and ability to speed up the feature delivery.

### **4.2 The Benefits to the Company**

- A better understanding of the cyber threat landscape and ability to prioritize corrective action.
- Quantified financial impact due to project security risks and ability to steer sales negotiations towards less risky
- Knowledge of the risk exposure and reduction in the residual risk after directed corrective action by the project management teams.
- Competitive advantage with the ability to show alignment with imperatives to "Build Security and Privacy in".

# References

## Acronyms

- SDLC: Software Development Life Cycle
- PM: Project Manager
- IT: Information Technology
- SOW: Statement of Work
- ISO: International Standards Organization
- CXO: Chief x Officer (X may be replaced by the initial letter for any of multiple corporate responsibilities)
- ISO27k: ISO 27001 – ISO defined standard for Information Security Management System (ISMS)
- ISMS: A company's Information Security Management System
- NCRs: Non-Compliance Reports created as a result of an audit showing non-compliance with the standard.
- OWASP: Open Worldwide Application Security Project
- SAMM: Security Assurance Maturity Model
- CIS: Center for Internet Security
- COTS: (for software applications) Commercial Of The Shelf
- NIST: National Institute of Standards and Technology
- BSIMM: Building Security In Maturity Model
- STRIDE: Acronym for the various malware impacts: Spoofing/ Tampering/ Repudiation/ Information Disclosure/ Denial of Service/ Elevation of Privilege
- FAQs: Frequently Asked Questions

# The Test SDK: The SDET's Best Friend in Monolith-to-Microservices Migrations

**Wayne Ng**

[wng@uship.com](mailto:wng@uship.com)

## Abstract

Transitioning from a legacy monolith to a distributed systems architecture presents both opportunities and challenges for the test automation engineer. It's exciting to use new modern test frameworks that are not bound by the confines of the legacy tech stack, but daunting at the same time given how easy it is for your test architecture to quickly become chaotic and unmanageable. How does one unify and standardize automation test architecture and functionality as more and more new services are spun up?

This article outlines uShip's approach to overcoming the hurdles involved in moving to distributed systems, how we developed an internal test software development kit to facilitate writing test automation and the lessons of how things have worked using the test software development kit (SDK) a few years later in our distributed system journey. It will also highlight how the SDK created an extensible strategy that enabled centralized methods of test data creation, standardized core test helpers, and allowed for simple distribution and updates for the engineering teams. Finally, we'll show how creating the SDK allowed engineering to easily roll out new future projects and improvements to our test automation in the ever changing evolution of our software.

## Biography

*Wayne Ng is currently a Principal Software Engineer in Test at uShip and has a decade of experience in the testing space, ranging from building test automation frameworks to establishing accessibility, chaos testing and monitoring best practices. He is a strong advocate about having a test automation first mindset to enable organizations to deliver value more frequently and more sustainably.*

*Outside of work, Wayne enjoys exploring nature, photography, eating good food and spending time with his wife and kids.*

## 1. Introduction

Migrating from a monolithic project to a distributed systems architecture sounds great on paper: deploys should go faster for individual components, ownership of code is clearer based on repos and there are opportunities to use new modern technologies. However, some things get lost in the transition because they may come as an architectural afterthought, such as a centralized testing framework familiar to everyone, a fully built-out library of page objects, and a variety of helper functions to facilitate navigating through flows or setting states for application components.

As functionality moves or new functionality is built outside of a monolith, different teams can easily start implementing core testing functionality in various ways or rewrite the same code too many times. For example, if your test needs a helper to set up a test user, no one wants the same user helper implemented 10 different ways, in 10 different places on 10 different teams. That's a lot of code changes for maintenance and updates if something systematically changes.

To prevent all chaos from breaking loose during a monolith to distributed architecture migration, this paper advocates for the planning and creation of a testing SDK as a non-negotiable, day-zero architectural activity, promoting a quality and automation first mindset.

## 2. The Benefits of the Test SDK

At its core, a testing SDK is a collection of tools and libraries designed to assist engineering in the process of testing their software. Generally, it contains functionality required for writing automated tests that can range from test data creation to fetching secrets or to setting configurations.

DRY (Don't Repeat Yourself) code is a central software development principle that allows for maintainable and scalable architectures. The primary goal of a test SDK is to prevent WET (Write Everything Twice) code since WET code makes updating code a nightmare. Test automation code is more often DAMP (Descriptive and Meaningful Phrases) so that it's easily understandable to see the behaviors the code is testing and sometimes allows for some repetition to make certain aspects of the user flow clear to the test engineer. A testing SDK is a way to help DRY that code out a little by reducing repetitive patterns and duplicate code and logic in favor of modular and referenceable code. In some ways, a testing SDK can be thought of as another manifestation of the factory pattern at an organization level scope - a single abstracted location where common functionality is maintained.

### 2.1 Centralized Methods For Leveraging Core Functionality

Pre-prepared, static datasets used in testing, also known as canned data, are not ideal when writing automated tests because the datasets need to be reset to a known good state prior to subsequent test runs. This can be problematic if a test fails since data might be in an unknown state prior to re-running the test, which may cascade more test failures. Tests with canned data often are slow and flaky as a result because they may need to run serially to prevent cross-contamination of the sample data store and ensure clean test runs.

In an ideal world, we would generate test data for setup in each test to maximize test isolation and test speed. New services spun off outside of a monolithic application are likely in service to that monolith, at least during the transition period. Tests will then share core business logic that is similar if not identical across the entire application.

For example, in an e-commerce application, you most likely have users and some form of a product, and those users interact with the product in some fashion. But if every microservice or single page application outside of the monolith needs a user and a product, in the spirit of DRY code, we do not want to write the same user or product creation script multiple times, especially if changes need to be made to users or the product.

Wrapping core API functionality in the test SDK solves much of the data creation problem. In doing so, there is a single place that will call that functionality, standardized such that API usages will be centralized. These tools can also be designed with flexibility in mind. If a test engineer needs the test data to be customized to meet their needs, they can easily pass in an override to the function signature to adjust the api call for their needs without the need for additional test code cruft.

```

    async function createUser(baseUser: UserModel, userOverwrites: UserOverwrites = {}): Promise<InternalUser> {
        const uuid = uuidv4().replace(/-/g, '');
        // create a deep copy of baseUser to avoid mutating it
        const baseUserCopy = JSON.parse(JSON.stringify(baseUser));
        // merge baseUserCopy with userOverwrites, including nested objects
        const model = {
            ...baseUserCopy,
            ...userOverwrites,
            user: {
                ...baseUserCopy.user,
                ...userOverwrites.user
            },
            userPreferences: {
                ...baseUserCopy.userPreferences,
                ...userOverwrites.userPreferences
            },
            profile: {
                ...baseUserCopy.profile,
                ...userOverwrites.profile
            }
        } as UserModel;

        // Update model with uuid
        model.user.userName = model.user.userName || `${baseUser.user.userId}${uuid}`;
        model.user.emailAddress = model.user.emailAddress || `${baseUser.user.userId}@node.uship.com`;

        const userInfo = await postUser(model);

        await patchTestUser(userInfo.userId, {
            EmailVerified: true
        });

        const internalIds = await getInternalUserInfo(userInfo.userId);

        return {
            ...userInfo,
            userInternalId: internalIds.data.userId,
            userGuid: internalIds.data.userGuid,
            isDeleted: internalIds.data.isDeleted
        };
    }

    export async function createShipper(userOverwrites?: UserOverwrites): Promise<InternalUser> {
        const shipper = await createUser(defaultShipper, userOverwrites);
        return shipper;
    }
}

```

Figure 1. Wrapper around user creation function with optional model overrides

Figure 1 above shows an example of how user creation is implemented in uShip's testing SDK. We have a base createUser function that takes in a defined typed userModel based on the types of users uShip has in its system and then a second parameter with optional user overrides to customize test data as needed. Once the desired user model is set up, the JSON payload is sent to the POST /users endpoint to create the user, followed by a subsequent PATCH operation for user property setup and then the user data is returned in the response. The second function in Figure 1 shows a simple implementation of how we use the generic createUser function to create a generic shipper user for reusable data creation that any test can call for setup given all our tests require a shipper.

## 2.2 Ability to Standardize Test Architecture

Working in a greenfield distributed system allows the opportunity for new technologies to become integrated into the tech stack since there are no legacy constraints if an engineering organization chooses to do so. While engineering teams often have freedom to use whatever tooling they wish, it is highly recommended to limit supported tooling during the distributed architecture migration to keep things maintainable. Standardizing the test framework allows teams to use a great new tool while keeping test patterns and writing styles uniform, so teams can ask one another questions and forge best practices within the supported technology stack. From an architectural perspective, an organization ideally wants to select one test framework. Just as an example in the Javascript space, no architecture group wants to support Playwright, Cypress, WebdriverIO, and Nightwatch all at the same time given they are all tools trying to solve the exact same problem.

As teams adopt core functionality within test frameworks, it's possible to bind the dependencies into the test SDK. While this does marry engineering to that test framework, it also allows QA to push many changes out across the organization in a simple fashion. Tightly coupling libraries into the test SDK comes with various pros and cons, so doing so may or may not be the right choice for organizations. Here are some of the benefits that we've seen as a result of binding a test framework to the test SDK.

```
export const desktopTest = baseTest.extend<DeviceTestFixture>({
    page: async ({ page, config }, use) => {
        const internalSdkConstants = await getConstants();
        await page.setExtraHTTPHeaders({
            'CF-Access-Client-ID': internalSdkConstants.cloudflareAccessClientId,
            'CF-Access-Client-Secret': internalSdkConstants.cloudflareAccessClientSecret
        });
        await page.route(`/${config.webUrl}`, async (route) => {
            await route.fulfill({
                headers: { 'access-control-allow-origin': '*' }
            });
        });
        await use(page);
    }
});
```

Figure 2: Playwright Fixture that is used across engineering for all end to end tests, regardless of repo

For example, in Figure 2 above as an example, uShip has chosen to adopt Playwright as its primary end-to-end test framework. To standardize our browser instances for the tests, we use Playwright's fixtures to set parameters such as viewport, device type, and browser type. For security reasons, every test in our organization also needs special headers appended to run in the VPC. Instead of having to manually add

them to every test in every repo, and updating them when the header secrets are updated, we include them in a base fixture. All our automation then inherits from that fixture in the test SDK.

A second benefit is the ability to roll out new initiatives within our QA organization through the test SDK. As an example, to roll out test automation analytics and metrics as a new project, we bind a single library to handle all labeling required for the metrics in the SDK. We can design it using interfaces so it can be consumed regardless of framework. This avoids having to change syntax between repositories because of implementation differences in the frameworks.

```
  export interface AllureReporter {
    // Include both method signatures to support allure wdio/jest/playwright
    label?(name: string, value: string): void;
    addLabel?(name: string, value: string): void;
}
```

Figure 3: An interface for Allure Reporting in the Test SDK

Figure 3 showcases an example interface for uShip's test metric reporter. Since test metrics are collected across various test frameworks at uShip, to unify metadata tagging, this interface in the test SDK allows us to use the same base library in the tagging function regardless of how the labeling function is implemented.

## 2.3 Simple Package Distribution for Engineering Teams

Once the library is ready to be packaged up and consumed by teams, the SDK gets published to our internal package repository. In a distributed architecture, some teams may install the testing SDK directly into their repo, or as new projects arise, another option is to have a boilerplate repository spun up that has the test SDK pre-installed as a dev dependency.

As updates are made to the test SDK, we use semantic release to push new versions to the package repository. Semantic release allows new changes to be version-controlled to avoid forcing breaking changes on the consumers of the SDK. This gives time for teams to choose when to update repositories to the latest release. If a team needs to consume new functionality immediately, they can simply just install the latest version of the package (i.e. bump from version 9 to version 10), but if the existing code needs some time to be updated due to a breaking change in the newest version of the package, the team can stay on the current version until refactoring and maintenance can be done on the repository.

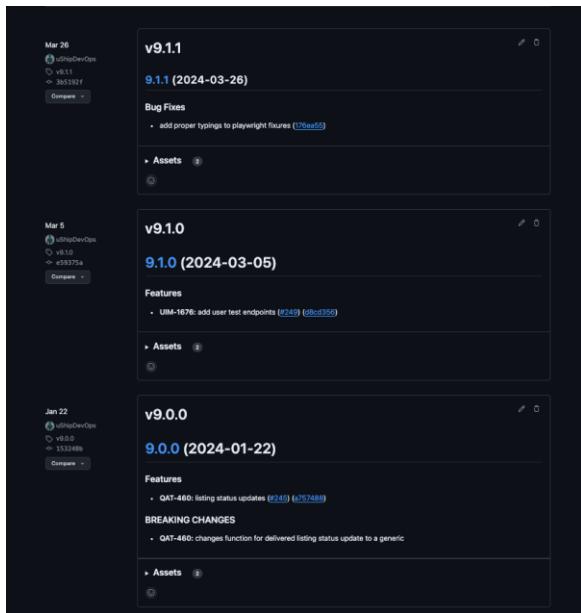


Figure 4: Test SDK Release Versions for a bug fix, feature release and a breaking change

Figure 4 above highlights example release versioning. Semantic release categorizes releases in three buckets: breaking changes (major version bump), new features (minor version bump) and bug fixes (minor version bump). A breaking change to the SDK necessitates changes by the consumer for the SDK to keep working. For example, renaming a function in the SDK would constitute a breaking change since the downstream consumer will need to call the newly named function for tests to keep working. Generally, adding any new functionality to the SDK will create a feature release. An example would be exposing a new endpoint under a namespace. An SDK bug fix would most likely come from a bug report by a consumer where an edge case of functionality is unexpected. For example, while the default function might work, adding an override causes a failure because a property is not respected accordingly and requires some fixes.

### 3. Tips and Tricks For Writing a Test SDK

A standardized, maintainable and easily distributed test SDK has many benefits as noted above but can be daunting to build from scratch. Below are some best practices for how to begin building and maintaining a test SDK.

#### 3.1 Start Your Test SDK By Exporting Existing Helpers

Most likely, somewhere in your legacy code, there is pre-existing test automation code and associated helper functions or API calls for data setup. When building out the test SDK, start by porting the core helpers needed in many tests, using the legacy code as the template to build out the SDK. Start with a MVP (minimal viable product) SDK and adjust the code as needed to make it flexible for necessary use cases. With version control, if you find a case that doesn't meet the requirements, you can always publish an update at a later time. If your distributed architecture is in the same programming language, this might be as simple as copy pasting code over to the new test SDK. Otherwise, translating legacy code into another language will require some effort but the core business logic and exposed api endpoints should hopefully already exist.

```

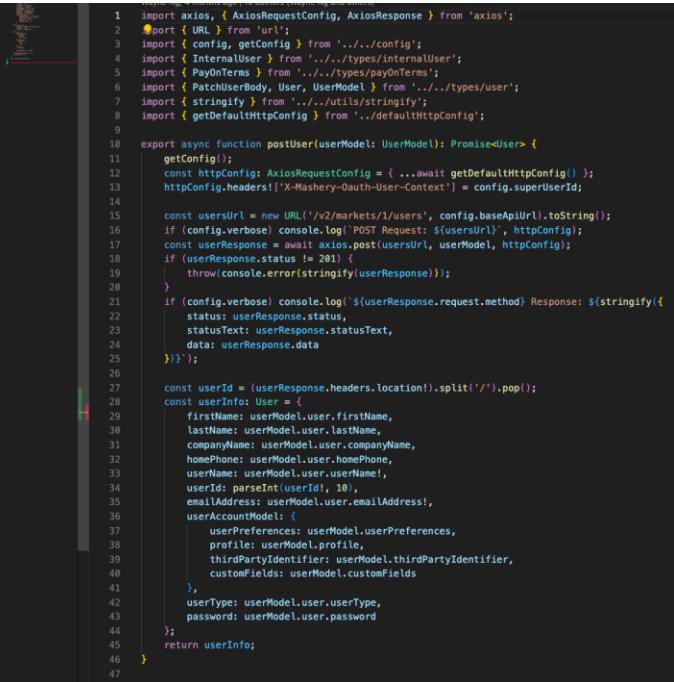
namespace uShip.EndToEndTests.Shared.Factories.Users
{
    public class Broker : User, IShipper, IServiceProvider
    {
        private Broker(UserModel userModel, InternalUserModel internalUserModel)
        : base(userModel, internalUserModel) {}

        public static async Task<Broker> CreateAsync<Action<PostAccountModel>>(Action<PerformDuringCreation> actionToPerformDuringCreation = null, [CallerMemberName] string caller = null)
        {
            var aboutMe = TestHelpers.GetTestName();
            var userModel = await PostUserModel.CreateAsync<PostAccountModel>(aboutMe, actionToPerformDuringCreation);
            // PostUserModel returns non-mutually held, so we will reactive them now
            await ActivateInternalUserModel(userModel);
            var internalUserModel = await GetInternalUserInfoAsync(userModel);
            return new Broker(userModel, internalUserModel);
        }

        protected static PostAccountModel GetAccountModel(string aboutMe)
        {
            var username = $"Broker-{Guid.NewGuid().ToString().Replace("-", "")}";
            return new PostAccountModel()
            {
                User = new PostUserModel
                {
                    UserType = "Broker",
                    Username = username,
                    FirstName = "FirstName",
                    LastName = "LastName",
                    EmailAddress = $"{username}@e.uship.com",
                    Password = "P@ssw0rd123456",
                    CompanyName = "Test Company",
                    HomePhone = "+1234567890",
                    PrimaryAddress = new PostAddressModel
                    {
                        StreetAddress = "123 Test Lane",
                        MajorMunicipality = "Austin",
                        PostalCode = "78701",
                        Country = "US"
                    },
                    AboutMe = aboutMe
                },
                UserPreferences = new PostUserPreferencesModel
                {
                    Currency = "USD",
                    TimeZone = "Central Standard Time",
                    SiteId = "UnitedStates"
                },
                Profile = new PostProfileModel
                {
                    BrokerNumber = "123456",
                    EntityType = 5,
                    TrailerTypes = new List<string> {"Reefer", "AirRideVan"},
                    Commodities = new List<string> {"Vehicles", "LTL", "Boats"},
                    Regions = new List<string> {"NA-USA"}
                }
            };
        }
    }
}

```

Figure 5: Existing legacy helper function for user creation (C#)



```

import { UserModel, UserType } from '../../../../../src/types/user';

const defaultBroker: UserModel = {
    user: {
        'userType': UserType.Broker,
        'userName': null,
        'firstName': 'FirstName',
        'lastName': 'LastName',
        'companyName': 'Test Company',
        'emailAddress': null,
        'password': 'P@ssw0rd123456',
        'homePhone': '+1234567890',
        'aboutMe': 'ScratchpadTest',
        'primaryAddress': {
            'streetAddress': '123 Test Lane',
            'majorMunicipality': 'Austin',
            'postalCode': '78701',
            'country': 'US',
            'stateProvince': 'TX'
        }
    },
    'userPreferences': {
        'timeZone': 'Central Standard Time',
        'currency': 'USD',
        'siteId': 'UnitedStates'
    },
    'profile': {
        'entityType': 5,
        'trailerTypes': [
            'Reefer',
            'AirRideVan'
        ],
        'commodities': [
            'Vehicles',
            'LTL',
            'Boats'
        ],
        'regions': [
            'NA-USA'
        ],
        'brokerNumber': '123456'
    },
    'thirdPartyIdentifier': null,
    'customFields': null
};

export default defaultBroker;

```

```

1 import axios, { AxiosRequestConfig, AxiosResponse } from 'axios';
2 import { URL } from 'url';
3 import { config, getConfig } from '../../../../../config';
4 import { InternalUser } from '../../../../../types/internalUser';
5 import { PayOnTerms } from '../../../../../types/payOnTerms';
6 import { PatchUserBody, User, UserModel } from '../../../../../types/user';
7 import { stringify } from '../../../../../utils/stringify';
8 import { getDefaultHttpConfig } from '../../../../../defaultHttpConfig';
9
10 export async function postUser(userModel: UserModel): Promise<User> {
11     const config = getConfig();
12     const httpConfig: AxiosRequestConfig = { ...await getDefaultHttpConfig() };
13     httpConfig.headers['X-Mashery-Oauth-User-Context'] = config.superUserId;
14
15     const usersUrl = new URL('/v2/markets/1/users', config.baseApiUrl).toString();
16     if (config.verbose) console.log(`POST Request: ${usersUrl}`);
17     const userResponse = await axios.post(usersUrl, userModel, httpConfig);
18     if (userResponse.status != 201) {
19         throw console.error(stringify(userResponse));
20     }
21     if (config.verbose) console.log(`User Response: ${stringify({
22         status: userResponse.status,
23         statusText: userResponse.statusText,
24         data: userResponse.data
25     })}`);
26
27     const userId = (userResponse.headers.location).split('/').pop();
28     const userInfo: User = {
29         firstName: userModel.user.firstName,
30         lastName: userModel.user.lastName,
31         companyName: userModel.user.companyName,
32         homePhone: userModel.user.homePhone,
33         userName: userModel.user.userName,
34         userId: parseInt(userId, 10),
35         emailAddress: userModel.user.emailAddress,
36         userAccountModel: {
37             userPreferences: userModel.userPreferences,
38             profile: userModel.profile,
39             thirdPartyIdentifier: userModel.thirdPartyIdentifier,
40             customFields: userModel.customFields
41         },
42         userType: userModel.user.userType,
43         password: userModel.user.password
44     };
45     return userInfo;
46 }
47

```

Figure 6: New version of helper function for user creation implemented in the test SDK (Typescript)

Figure 5 and Figure 6 above show an user creation function that has been ported from the legacy monolithic test helpers in C# and rewritten in Typescript in the test SDK (split into 2 files). Most of the logic used in the creation of users remains unchanged, with just some minor adjustments for semantics that have evolved over the 20 years of our system and differences between C# and Typescript. Porting code into a

test SDK, whether it's in the same programming language or different, can be done fairly quickly if there are no major updates that need to be made. Especially if your organization allows use of generative AI such as chatGPT, code can be very quickly transcribed if porting languages and allows for rapidly building out the test SDK.

### **3.2 Limit SDK Contents to Reusable Functionality**

To prevent having a massive monolithic test SDK, it is advised to only put functionality in the SDK that will be used by consumers in multiple projects (i.e. more than one). Most likely, this functionality will be core business logic that most teams will need to implement in their tests. For example, almost all companies likely have some concept of a user, so standardizing and centralizing user creation in a test SDK would be a good starting point.

One-off functionality in the SDK is not helpful if no one else intends on consuming it, and thus can just live in the project where it is needed as a traditional helper function. As organizations discover that functionality is no longer a one-off, that would create an opportunity to add the functionality to the SDK and consume the newest version of the SDK. Collaboration between teams is key to finding out when one-off functionality becomes used in multiple places.

### **3.3 Valid Table of Contents**

As new functionality gets added to the SDK, make sure the README or API documentation for your SDK is up to date. Especially as fixes and changes are made, having up-to-date documentation greatly aids understanding for engineers who may not be as familiar with the SDKs. Some plugins will auto-generate a table of contents to speed up documentation. Also, having sample implementations is beneficial so engineers can quickly either copy existing calls or understand how the functions work.

**Configuring internal-sdk**

Code consuming the `internal-sdk` must configure which environment to use. You may override properties as needed. Configuration examples for dev:

```

const uship = require("@uship/internal-sdk");
// By default, the configuration for the sdk is set for
// ushipdev.com
uship.configure();

// Can pass in optional env parameter for other envs
uship.configure("qa");

// Can override default properties
uship.configure("qa", {
  verbose: true,
});

```

Configuration object properties

Property	Type	Default	Description
<code>env</code>	<code>string</code>	<code>dev</code>	the environment this config is set up for
<code>webUrl</code>	<code>string</code>	<code>https://www.foobar.com</code>	the base url of the uship web application
<code>lambdaUrl</code>	<code>string</code>	<code>https://www.foobar.com</code>	the url of the lambda to test against for PRs, defaults to cloudflare domain url if none provided
<code>baseApiUrl</code>	<code>string</code>	<code>https://api.foobar.com</code>	the base url of the uship api
<code>superUserId</code>	<code>number</code>	<code>1</code>	the generated id of the super user
<code>adminId</code>	<code>number</code>	<code>10</code>	the id of an admin user
<code>verbose</code>	<code>boolean</code>	<code>false</code>	whether or not the internal-sdk will log api calls to the console

Figure 7: SDK Documentation for how to configure the SDK

Figure 7 above shows sample documentation in the test SDK README file. This sample function configures uShip's test SDK so that individual repos can point their tests to run against any environment, which can help validate functionality that has environment parity in the system. Ideally, the README should have a high level description of what the function does, sample code that implements the function and property information for any parameters.

### 3.4 Strongly Typing Your Test SDK

Using a strongly typed language such as Java or Typescript to build your test SDK is highly recommended. A strongly typed language is one in which types are enforced strictly by the compiler or inter-

prefer so that each variable is bound to a specific data type, and operations on these variables are restricted to what is allowed for their data type. Attempting to perform operations that are not allowed for a particular type, such as adding a string to an integer without explicit conversion, will result in a type error which will help minimize bugs in the test automation code. For example, presume your user creation API is not well documented and a new engineer is unsure whether an override to customize the user takes a string or an integer or some other complex object parameter. While it would be ideal to have API documentation, the strongly typed test SDK would error and return the error code that would help the engineer realize their mistake.

Using a strongly typed language also allows autocomplete when using the test SDK to quickly fill in functionality in tests. Especially with AI tooling like Github Copilot available in some engineering organizations, this can greatly speed up test writing efficiency. It also prevents misuse of the SDK by enforcing the data types, helping data remain close to its production state if modifications need to be made and mirror the end user experience.

### **3.5 Testing Your Test SDK**

The testing SDK should be treated as production code, conforming to your organization's software development best practices. Changes to the SDK should go through the same SDLC process that other code changes follow (code reviews, QA, etc). While much of the testing SDK is designed for convenience to make testing easier for engineers, it should also be designed in a way that doesn't compromise security (i.e. do not directly expose API keys or secrets). It is highly recommended to write test automation (unit, end to end tests) using the functions in the test SDK to ensure they work as expected as part of the SDK's own deployment pipeline.

### **3.6 Feature Requests**

As the business builds new features, likely new functionality will need to get added to the test SDK or changes in business logic cause breaks in the SDK. Having a process to handle necessary new feature requests or bug tickets aids the maintainer of the package to know what priority to work on necessary changes. uShip uses Github Issues to track feature requests and bugs for the test SDK with associated tickets on a Kanban board. This way, changes to the test SDK can go through the engineering organizations normal prioritization and SDLC workflow.

## **4. The Challenges of the Test SDK**

Like many other projects in an engineering organization, the hardest aspect of having a test SDK is its maintenance. Since much of the SDK houses wrappers around common shared functions, making a breaking change to the APIs associated with SDK functionality means remembering to also change the SDK accordingly. Failure to do so could break future SDK builds or potentially disrupt pipelines in the distributed system.

General maintenance is another challenge if the SDK lacks clear ownership. Since the SDK includes functionality from different engineering domains, feature ownership is shared. Adding a CODEOWNERS file can help enforce code reviews for pull requests and establish ownership. At uShip, the QA group, and specifically SDETs own the test SDK. Even with ownership set for an SDET group, a cadence needs to be established to review dependencies, security, and to apply fixes or updates.

Setting up a bot such as Renovate or Dependabot helps alleviate some maintenance but still requires diligence to merge in all generated pull requests. Proper teardown and disposal of test data after test runs is critical, depending on the data cleanup maturity of your engineering organization, to maintain a manageable sized data store. Work with your data team to determine a strategy for cleaning up the test data prior to implementing the test SDK.

Maintenance on a consumer level across engineering is another challenge. When new major versions are released or old versions need to be deprecated, engineering teams must find bandwidth to update the packages and affected tests. While teams are often content with using versions that meet their needs, security updates or other critical changes may require significant investment to update to the latest version. For example, at uShip, we have some teams still on v.6.0.0 of our test SDK when the latest version of the test SDK is on v.10.0.0, which means several teams are months, if not years, behind the most recent version. This requires the SDET group to continue supporting v.6.0.0 of the SDK until teams can update their code to use v.10.0.0.

## 5. Conclusion

This paper highlights the importance of having a testing SDK as testing gets increasingly decentralized in a distributed system. By creating a test SDK, a quality team is able to wrap core business APIs in a single maintainable package for test data generation, enforce standardized test architecture to remain scalable as the application grows and simplify distribution as changes to the APIs get made. Especially as an organization scales, having a testing SDK reduces the stress of having to build additional test infrastructure in new projects since the core helpers all exist in a single package. While building and maintaining the testing SDK requires effort and resources, the rewards enable a significantly more manageable test architecture that accelerates feature delivery to end-users while empowering QA to confidently write tests.

## 6. References

dbt. "DRY." Analytics Engineering Glossary. <https://docs.getdbt.com/terms/dry>. (accessed May 10, 2024).

Martin Fowler. "Page Object," martinfowler.com, <https://martinfowler.com/bliki/PageObject.html>. (accessed May 10, 2024).

Microsoft. "Fixtures." Playwright. <https://playwright.dev/docs/test-fixtures>. (accessed May 10, 2024)

Semantic Release. "Semantic Release," Github, <https://github.com/semantic-release/semantic-release> (accessed May 10, 2024).

Renovatebot. "Renovate Documentation," Renovate Docs, <https://docs.renovatebot.com/> (accessed May 10, 2024).

## ***Exceeding Expectations, Inspiring Quality***

**Nishadhi Nikalandawatte**

<https://www.linkedin.com/in/nishadhinikalandawatte/>  
[nishadhin@slalom.com](mailto:nishadhin@slalom.com)

**Andy Jimenez**

<https://www.linkedin.com/in/andy-jimenez-36992b/>  
[andyj@slalom.com](mailto:andyj@slalom.com)

### **Abstract**

In the realm of organizational excellence, the concept of quality stands as a pillar of success, yet its interpretation and execution often pose significant challenges. Many organizations traditionally define quality as the strict adherence to specifications, inadvertently overlooking the nuanced processes that underpin true excellence. This narrow definition fails to account for the intricate journey towards achieving quality outcomes. The consequences of overlooking quality can be staggering, ranging from substantial financial losses to irreparable damage to a company's reputation and trust. Drawing from our collective experiences, we aim to illuminate the path towards cultivating a Quality Mindset within organizations of all sizes. In this comprehensive exploration, we seek to demystify the multifaceted nature of quality and offer actionable strategies to bridge organizational gaps, inspiring a culture of enduring excellence.

### **About the Authors**



Nishadhi is an accomplished Software QE Strategist with 20+ years of industry experience. She has extensive experience in building automation frameworks and providing guidance on creating End-To-End test solutions for customers. Thriving off working with other people, she has a strong passion for advocating for quality in organizations, coaching, mentoring, and building out high performing teams.



Andy has worked within corporate IT departments, Research & Development, as well as consulting services in a career that spans over 20 years in the technical industry. He has worked across many different technologies, platforms, and on infrastructure that served over 2 million users per day. Combining his passions for quality and for people, he uses his vast experience to help level up Quality programs for organizations.

## 1. Introduction

Embarking on a journey toward organizational excellence, this paper presents a compelling narrative on the essence of quality in the business sphere. Quality, a term often confined within the boundaries of compliance, unfolds here as a multidimensional concept that drives success and fosters sustainable growth. We begin by dissecting the term "Quality," revealing its subjective nature and the diverse factors that shape its understanding across industries, stakeholders, and cultures. The paper then introduces the transformative "Quality Mindset," a paradigm shift that prioritizes quality at every organizational level, advocating for customer focus, continuous improvement, and innovation.

As we navigate through the intricacies of quality assessment, we address the pivotal question of identifying quality gaps, highlighting the importance of aligning documented procedures with actual practices and utilizing quality metrics to unveil areas for enhancement. The discussion extends to the dire consequences of neglecting quality, underscoring the risks of inconsistency, customer dissatisfaction, and financial losses that can ensue.

In a decisive move towards rectification, we explore the implementation of the "Quality Mindset," detailing the roles and responsibilities of executive leadership, program management, and individual teams in fostering a culture of excellence. The paper concludes by tackling the challenges inherent in this cultural shift, offering strategies to overcome resistance and sustain a commitment to quality that resonates with the organization's strategic objectives.

Join us as we delve into these themes, advocating for a robust governance framework and a steadfast dedication to quality that transcends mere metrics, embodying the core of organizational identity and success.

## 2. Defining "Quality"

So, how do **you** define Quality?

Quality embodies the fusion of precision, reliability, and excellence across all aspects of products, services, and processes. Beyond rigid adherence to predetermined criteria, quality encompasses the meticulous processes that underpin consistent excellence. At its core, quality reflects a steadfast commitment to innovation, integrity, and continuous improvement, driving organizations towards unparalleled success and distinction in an ever-evolving landscape.

There are several reasons why there can be different understandings of quality, and these variations often stem from diverse perspectives, contexts, and priorities. Some key factors contributing to different interpretations of quality are as follows:

- **Subjectivity** - Views on quality can vary greatly and are often shaped by personal tastes, past encounters, and viewpoints. What someone deems as excellent might not hold the same value for another. Take clothing, for instance: while some prioritize visual appeal and

opt for fast fashion, others prioritize longevity and seek garments that will withstand the test of time.

- **Stakeholder Differences** - Various stakeholders, such as customers, producers, and regulators, often harbor unique priorities and anticipations concerning quality. These differing viewpoints can result in diverse interpretations of what qualifies as a top-notch product or service. For instance, regulators may emphasize compliance with regulatory standards, while customers may prioritize usability and user experience.
- **Industry Standards** - Various sectors may uphold distinctive norms and benchmarks for assessing quality. What constitutes excellence in one industry may not hold sway or relevance in another. For instance, the International Organization for Standardization (ISO) establishes quality control standards tailored for manufacturing enterprises to follow, yet these might not be directly applicable to sectors such as healthcare or automotive industries.
- **Cultural Variations** - Cultural differences can influence perceptions of quality. Preferences, values, and expectations regarding products or services may vary across cultures, leading to different understandings of quality.
- **Functional vs. Aesthetic Quality** - Some individuals may prioritize functional aspects of quality (e.g., performance, reliability), while others may place more emphasis on aesthetic qualities (e.g., design, appearance). These differing priorities contribute to diverse perspectives on quality.
- **Evolving Standards** - Standards of quality can change over time due to technological advancements, shifts in consumer expectations, or changes in industry practices. This evolution can result in different understandings of quality at different points in time.
- **Economic Considerations** - Economic factors, such as cost constraints and market competition, can impact perceptions of quality. In some cases, a lower price point may be associated with lower quality, while in others, cost-effectiveness may be a key determinant of quality.
- **Contextual Relevance** - The environment in which a product or service is utilized or experienced can shape how quality is perceived. What may be deemed excellent in one setting might not be appropriate or applicable in another. For instance, a data-capturing product may necessitate varying degrees of accuracy depending on its application (such as the quality required for social media videos versus that needed for remote surgical procedures).
- **Education and Awareness** - Differences in education levels and awareness about certain products or services can contribute to variations in understanding quality. Informed consumers may have different expectations compared to those who are less knowledgeable.

**"Quality means doing it right when no one is looking."**

**-Henry Ford**

True quality demands unwavering dedication to doing things correctly, regardless of external scrutiny. It speaks to the character of individuals and organizations, highlighting their commitment to upholding high standards even in the absence of accountability. This mindset fosters a culture of excellence where every task, whether big or small, is approached with diligence, attention to detail, and a relentless pursuit of perfection. Ultimately, true quality is not simply a checkbox to be ticked off in the presence of others, but is a guiding principle that shapes actions and decisions even in solitude.

While each factor contributes uniquely to the overall understanding and assessment of quality, their significance and impact may vary depending on the specific context and circumstances. Subjectivity and stakeholder differences influence individual perceptions and expectations, while industry standards provide a common framework for assessment. Cultural variations shape preferences and values, influencing aesthetic and functional quality considerations. Evolving standards reflect ongoing improvements and advancements, while economic considerations and contextual relevance play roles in determining value and applicability. Education and awareness initiatives foster understanding and appreciation of quality, while regulatory requirements ensure compliance and safety. Collectively, these factors contribute to the multifaceted nature of quality, requiring a nuanced approach to assessment and management to meet diverse stakeholder needs and expectations.

### 3. Defining the "Quality Mindset"

A **Quality Mindset** is the collective embodiment of attitudes, beliefs, and behaviors that elevate the importance of quality in every facet of work and life. It's a cornerstone in business, product development, project management, and other domains where excellence is non-negotiable. This mindset is the heartbeat of a culture that thrives on continuous improvement and unwavering customer satisfaction.

#### **Core Characteristics of a Quality Mindset:**

- **Customer Focus** - The cornerstone of a quality mindset is the unwavering commitment to satisfy and exceed customer expectations, catering to both internal stakeholders and external clients.

- **Continuous Improvement** - Embracing the idea that there is always room for improvement and actively seeking ways to enhance processes, products, and services.
- **Holistic Perspective** - Being meticulous and thorough in work, paying attention to the finer details to ensure that the result is of high quality.
- **Prevention Orientation** - Focusing on preventing errors and issues rather than simply fixing problems after they occur. This entails taking proactive steps to detect and mitigate potential issues before they escalate. For instance, if you're a quality engineer, your emphasis shouldn't be solely on identifying bugs; rather, it should be on the actions taken to avert severe bugs in both testing and production environments.
- **Ownership and Accountability** - Taking ownership of one's work and being accountable for the quality of the outcomes. This includes learning from mistakes and actively working to rectify them.
- **Innovation and Creativity** - Quality-minded individuals and organizations are often more open to innovation and creativity. They seek better ways of doing things, which can lead to the development of new products, services, or processes over the long term.
- **Clear Standards and Processes** - Establishing clear standards, guidelines, and processes that contribute to a consistent and reliable approach to achieving quality objectives.
- **Do the right thing** - Making decisions and taking actions that prioritize and uphold high standards of quality.

At its core, a **Quality Mindset** cultivates an environment of excellence, where every individual and team are aligned with the mission of delivering superior outcomes. It's not just a practice but a principle that propels organizations to flourish in a competitive and ever-changing marketplace.

### **It takes a village.**

Cultivating a quality mindset is important for organizations seeking to establish a reputation for excellence and reliability. It involves instilling a shared commitment to quality throughout the organization, from leadership to individual contributors.

### **The Journey to a Quality Mindset:**

Expanding on the focal points highlighted earlier, it becomes apparent that transitioning to a quality mindset necessitates a comprehensive organizational transformation. At any organization, we have the opportunity and responsibility to guide our clients through this transformative journey and elucidate its intricacies. Implementing facets of the Quality Mindset entails more than just surface-level adjustments; it demands a concerted effort and robust governance framework to ensure success. This process is far from minor; it represents a fundamental shift in organizational culture and operations. As with any significant change, it will require ongoing, diligent adjustments to strategies, approaches, and communication channels to effectively embed the principles of quality across all levels of the organization.

### **3.1 How to Identify Quality Gaps?**

Identifying quality assurance gaps in an organization requires a comprehensive approach, encompassing diverse aspects of the quality management process. One avenue involves reviewing documented procedures and comparing them against actual practices to pinpoint discrepancies or areas lacking clear guidelines. For instance, while a software development company may have robust protocols outlined, closer inspection might reveal inconsistencies in their execution by team members, leading to testing process disparities. Additionally, analyzing quality metrics like defect rates and customer satisfaction scores can unveil shortcomings in meeting standards. For example, persistent high defect rates in a specific product line for a manufacturing company may signal production process deficiencies. Conducting regular audits of quality procedures and practices is another effective strategy, uncovering non-compliance issues or control measure deficiencies. By implementing these approaches and addressing the identified gaps, organizations can enhance their practices and ensure high-quality product delivery. The following diagram illustrates methods for identifying Quality Gaps.

**"Be a yardstick of quality. Some people aren't used to an environment where excellence is expected."**  
-Steve Jobs



By meticulously implementing these strategies, organizations can systematically identify, thoroughly analyze, and promptly address any quality engineering and assurance gaps present within their operations. This comprehensive approach enables them to implement targeted enhancements, refine existing processes, and institute robust quality control measures, thereby enhancing the overall quality and reliability of their products and services to meet and exceed customer expectations.

### 3.2 What happens if Quality is not a priority of an organization?

Neglecting quality within an organization can trigger a cascade of adverse consequences. Failing to prioritize quality can result in products or services riddled with inconsistencies, unreliability, and an inability to meet customer expectations, ultimately leading to dissatisfaction and a loss of trust, which in turn can lead to missed business opportunities. Poor quality can also incur additional costs through rework, returns, and warranty claims, while simultaneously tarnishing the organization's reputation, eroding its competitive advantage, and demoralizing employees. Furthermore, non-compliance with quality standards exposes the organization to legal risks and operational inefficiencies, highlighting the critical importance of embedding quality as a core value and operational principle to safeguard customer satisfaction, financial performance, reputation, and employee morale. The following image illustrates the ramifications when quality is not a priority in an organization from five different perspectives.

#### **When quality is NOT a priority...**

*Organizations are at risk of encountering numerous challenges and negative outcomes that can impede their success and long-term sustainability.*

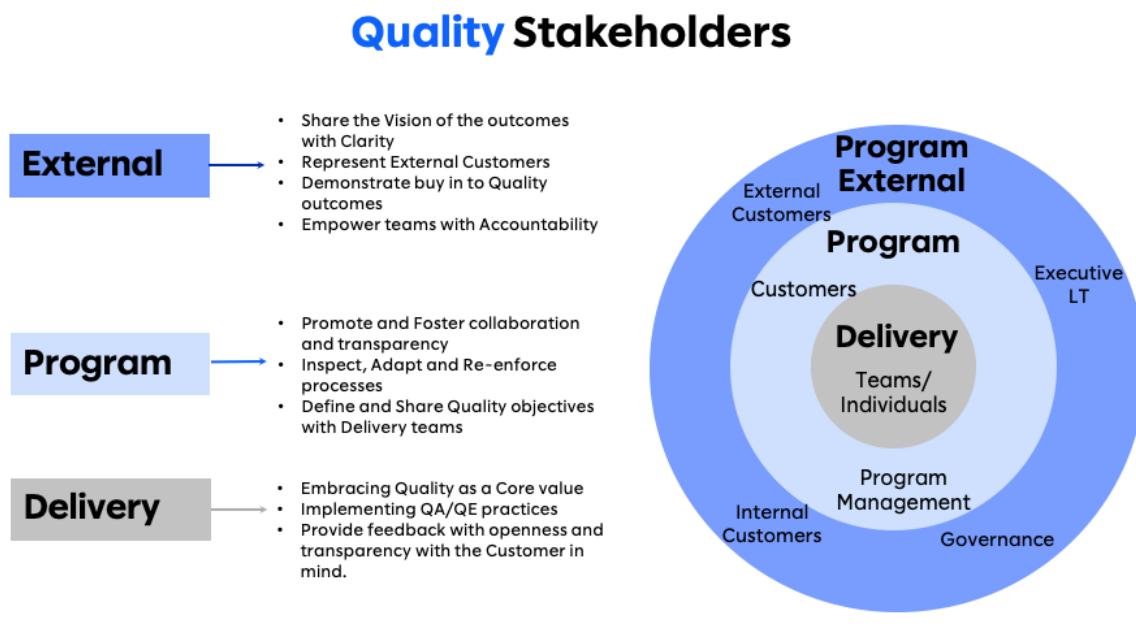
DECREASED CUSTOMER SATISFACTION	LACK OF ACCOUNTABILITY	LACK OF PRODUCTIVITY	ABSENCE OF LONG-TERM VISION	LEGAL AND COMPLIANCE RISK
<ul style="list-style-type: none"> <li>✓ Dissatisfied customers are less likely to make repeat purchases or recommend products or services to others, leading to a decline in sales and revenue.</li> <li>✓ Loss of customer loyalty.</li> <li>✓ Damage to the organization's reputation and credibility in the market.</li> <li>✓ Dealing with complaints, returns, and refunds from dissatisfied customers can increase customer service costs and strain resources.</li> </ul>	<ul style="list-style-type: none"> <li>✓ Internal disarray, confusion, and inefficiency within the organization.</li> <li>✓ Struggles by teams to meet customer expectations.</li> <li>✓ Challenges in aligning with quality objectives.</li> <li>✓ Quality treated as a finger pointing or blaming game.</li> <li>✓ Without clear accountability, quality issues may go unresolved or unaddressed, leading to persistent defects, errors, or inconsistencies in products or services.</li> </ul>	<ul style="list-style-type: none"> <li>✓ Impacted organizational performance and competitiveness.</li> <li>✓ Inefficient processes lead to delays, errors, and increased costs.</li> <li>✓ Low productivity can result in employee frustration and demotivation.</li> <li>✓ Hindered ability to meet customer expectations and deliver high-quality products or services on time.</li> </ul>	<ul style="list-style-type: none"> <li>✓ Inconsistency and diminished competitiveness.</li> <li>✓ Short-term fixes take precedence over sustainable quality management strategies.</li> <li>✓ Innovation opportunities are missed due to the absence of long-term planning.</li> <li>✓ The organization falls behind competitors and struggles to achieve strategic objectives.</li> <li>✓ Increased costs for the organization in terms of time, resource and money.</li> </ul>	<ul style="list-style-type: none"> <li>✓ Non-compliance with quality standards, regulations, or industry best practices can lead to legal liabilities, lawsuits, fines, or penalties.</li> <li>✓ Legal battles and regulatory fines can cause significant financial losses, including legal fees, settlement costs, and damage awards.</li> <li>✓ Disruption to normal business operations, diverting resources and attention away from core activities and strategic initiatives.</li> <li>✓ Persistent non-compliance may lead to market exclusion as customers and partners prefer organizations that prioritize and adhere to quality requirements.</li> </ul>

To ensure long-term success and mitigate associated risks, organizations must prioritize quality as a fundamental aspect of their operations and strategic objectives. This involves investing in quality management processes, cultivating a culture of continuous improvement, and emphasizing the importance of quality across the organization. By doing so, businesses can establish trust with

customers, bolster their reputation, and foster sustainable growth. Prioritizing quality enables organizations to deliver value to customers, adapt to evolving market dynamics, and maintain a competitive advantage. In the following sections, we will delve into implementing the quality mindset within an organization, outline key stakeholder responsibilities, and address the challenges that may arise during the process.

## 4. Implementing the “Quality Mindset”

To effectively implement Quality initiatives through the adoption of the Quality Mindset, the collaboration and participation of diverse stakeholders are imperative to achieve the desired outcomes. The successful realization of these initiatives hinges upon the concerted efforts and contributions of various stakeholders across different levels of the organization. The following diagram delineates a comprehensive overview of the overarching objectives and responsibilities assigned to each stakeholder, guiding their roles and accountabilities in driving the organization towards a culture of continuous improvement and excellence.



### 4.1. Navigating the Quality Landscape: Executive Leadership

In the quest for organizational excellence, the role of executive leadership is paramount. These leaders are the architects of a quality-centric culture, laying the foundation for a resilient and dynamic organization. Their commitment to quality is not just a statement of intent; it reflects the organization's core values and strategic direction. Here are some key responsibilities they typically take on:

- **Setting the Vision** - Executive leaders need to define and communicate a clear vision of what quality means for the organization. This includes setting quality objectives, outlining the benefits of a quality mindset, and emphasizing its importance in achieving overall business goals.
- **Allocating Resources** - They allocate resources, including budget, time, and personnel, to support quality initiatives. This might involve investing in training programs, quality tools and technologies, and hiring skilled professionals to drive quality improvement efforts.
- **Establishing Policies and Processes** - Executive leaders develop policies and processes that promote a culture of quality throughout the organization. This includes implementing quality standards, defining best practices, and creating frameworks for continuous improvement.
- **Leading by Example** - They demonstrate their commitment to quality through their actions and decisions. By prioritizing quality in their own work and interactions, they set a powerful example for others to follow.
- **Providing Support and Guidance** - Executive leaders support quality champions and teams by providing guidance, removing obstacles, and offering encouragement. They serve as mentors and coaches, helping individuals and teams develop the skills and confidence needed to excel in quality-related roles.
- **Monitoring Performance** - They establish metrics and KPIs to assess the effectiveness of quality initiatives and track progress over time. By regularly reviewing performance data, they can identify areas for improvement and make informed decisions to drive continuous quality enhancement.
- **Celebrating Successes and Recognizing Contributions** - Executive leaders celebrate achievements and recognize the contributions of individuals and teams who demonstrate a commitment to quality. This helps reinforce positive behaviors and fosters a culture of appreciation and recognition.
- **Driving Continuous Improvement** - Finally, they are responsible for driving a culture of continuous improvement, where quality is not seen as a one-time initiative but as an ongoing journey. This involves fostering innovation, embracing feedback, and adapting to changing market dynamics and customer needs.

When executive leadership wholeheartedly commits to these roles, the organization is galvanized to embrace a quality mindset as a fundamental principle. This mindset becomes the lens through which every decision is made, and every action is evaluated, ensuring that quality is woven into the fabric of the organization's operations. It is through this steadfast dedication to quality that an organization not only achieves its goals but sets new benchmarks for excellence in its field.

## 4.2. Cultivating Excellence: The Strategic Influence of Program Management

In the tapestry of organizational success, program management teams are the weavers of quality, meticulously integrating a quality mindset into the very threads of the organization's fabric. Their role is not just supportive but strategic, as they align the sails of quality initiatives with the winds of the organization's overarching goals. Here's a closer look at their pivotal responsibilities:

- **Alignment with Organizational Goals** - Program managers should ensure that quality initiatives and activities align with the organization's overall strategic goals and objectives.

They work closely with executive leadership to understand the strategic vision and translate it into actionable quality plans.

- **Quality Plan Development** - Program managers develop comprehensive quality plans that outline the strategies, objectives, and activities necessary to promote a culture of quality. These plans often include goals for process improvement, defect reduction, customer satisfaction, and other key quality metrics.
- **Talent Management** - They are responsible for allocating resources, including budget, personnel, and technology, to support quality improvement initiatives. This may involve coordinating with various stakeholders to ensure that the necessary resources are available and effectively utilized.
- **Stakeholder Engagement** - Program managers engage with stakeholders across the organization to build consensus and support for quality initiatives. This includes communicating the benefits of a quality mindset, addressing concerns, and soliciting feedback to ensure that quality objectives are aligned with stakeholder needs and expectations.
- **Risk Management** - They identify potential risks and barriers to quality improvement and develop strategies to mitigate them. This may involve conducting risk assessments, implementing preventive measures, and establishing contingency plans to address unforeseen challenges.
- **Monitoring and Reporting** - Program managers monitor the progress of quality initiatives and regularly report on key performance indicators (KPIs) and metrics to stakeholders. This helps track the effectiveness of quality efforts, identify areas for improvement, and make data-driven decisions to drive continuous improvement.
- **Facilitating Collaboration** - They facilitate collaboration and communication among cross-functional teams involved in quality improvement efforts. This may include organizing meetings, workshops, and training sessions to share best practices, lessons learned, and success stories.
- **Driving Accountability** - Program managers hold team members and stakeholders accountable for their roles and responsibilities in promoting a culture of quality. This may involve establishing performance expectations, providing feedback, and recognizing achievements to motivate continued commitment to quality improvement.

By embracing these multifaceted roles, program management becomes the catalyst for a transformative quality mindset, fostering an environment where excellence is not an aspiration but a reality. Their strategic influence extends beyond immediate tasks, seeding a legacy of improved products, elevated customer satisfaction, and enhanced business performance. In the grand narrative of organizational achievement, the chapters authored by program management are pivotal, guiding the story toward a future where quality is the protagonist, celebrated and sustained across every level of the enterprise.

#### 4.3. Quality at the Helm: The Vital Role of Teams and Individuals

In the intricate dance of organizational performance, teams and individuals hold the spotlight, their every step resonating with the rhythm of quality. They are the standard-bearers, carrying the banner of excellence in each task and decision. Here's a glimpse into the key roles they play in this grand performance:

- **Embracing Quality as a Core Value** - Team members must embrace quality as a fundamental value in their work. This involves understanding the importance of delivering high-quality products or services and taking pride in their craftsmanship.
- **Understanding Requirements and Expectations** - Teams need to thoroughly understand the requirements and expectations of customers, stakeholders, and other relevant parties. This includes understanding quality standards, specifications, and performance criteria that need to be met.
- **Contributing to Process Improvement** - Teams play a crucial role in identifying opportunities for process improvement and implementing solutions to enhance quality and efficiency. This may involve suggesting changes to workflows, procedures, or tools to streamline processes and reduce errors.
- **Taking Ownership of Quality** - Team members should take ownership of the quality of their work and strive for excellence in everything they do. This includes taking pride in delivering defect-free products or services and taking corrective action when issues arise.
- **Providing Feedback and Suggestions** - Team members should provide constructive feedback and suggestions for improving quality-related processes, tools, and methodologies. This feedback helps identify areas for improvement and drive positive change within the organization.
- **Being Proactive In Problem-Solving** - Teams should be proactive in identifying and addressing quality issues as they arise. This may involve troubleshooting problems, implementing corrective actions, and preventing recurrence of issues in the future.
- **Executing Quality Assurance Activities** - Teams are responsible for executing quality assurance activities to ensure that products or services meet established quality standards and requirements. This may include conducting inspections, reviews, and tests at various stages of the development or production process.
- **Championing a Culture of Quality** - Finally, teams should champion a culture of quality within their organization by promoting awareness, recognition, and celebration of quality achievements. This helps foster a positive environment where quality is valued and prioritized by everyone.

As the curtain falls on each project, the contributions of teams and individuals shine through, their commitment to quality leaving an indelible mark on the organization's legacy. It is through their unwavering dedication that the organization not only meets but exceeds expectations, securing customer satisfaction and charting a course for enduring success. In this collective endeavor, quality is not just a goal but a journey, one that is continuously enriched by the collective efforts of each member, driving the organization to new heights of excellence.

## 5. Overcoming Challenges in Implementing a Quality Mindset

Implementing a Quality Mindset within an organization is a transformative journey marked by numerous challenges. Key obstacles include employee resistance to change, lack of executive support, resource constraints, and communication barriers. Overcoming these challenges necessitates strong leadership, allocation of resources, and a commitment to breaking down silos and fostering cross-functional teamwork. Additionally, investing in skills development and

streamlining processes can enhance the successful implementation of a quality mindset. Despite pressures to prioritize short-term goals, organizations must maintain a balance between immediate needs and long-term considerations of quality and customer satisfaction. External factors such as market competition and regulatory requirements further complicate this balance, requiring careful management. Ultimately, implementing a quality mindset is an ongoing journey that demands dedication, perseverance, and continuous reinforcement to sustain momentum and drive improvement.

## Challenges of QM implementation

Here are some common obstacles that organizations may encounter. Overcoming these challenges demands organizational commitment, cultural change, and ongoing investment. Ultimately, success breeds a culture where quality is ingrained at all levels.

	<b>Resistance to Change</b>		<b>Lack of Stakeholder Buy-In</b>
	<b>Limited Resources</b>		<b>Communication Barriers</b>
	<b>Inadequate Skills and Expertise</b>		<b>Complexity of Processes</b>
	<b>Resistance to Quality Metrics</b>		<b>Short-Term Focus</b>
	<b>External Pressures and Market Dynamics</b>		<b>Sustainability of Efforts</b>

### ***Addressing Quality Mindset Implementation Challenges involves several key steps:***

- **Identify Challenges** - Begin by identifying the specific challenges or barriers to implementing a quality mindset within your organization. This may involve conducting surveys, interviews, or workshops to gather insights from employees at all levels.
- **Create Awareness** - Increased awareness among employees about the importance of a quality mindset and its impact on organizational success. This can be achieved through training sessions, workshops, and internal communications emphasizing the benefits of quality-driven practices.
- **Provide Training and Resources** - Offer training programs and resources to equip employees with the knowledge, skills, and tools necessary to adopt and maintain a quality mindset. This may include training on quality management principles, problem-solving techniques, and quality improvement methodologies.
- **Establish Clear Expectations** - Set clear expectations for employees regarding their roles and responsibilities in upholding a quality mindset. This includes defining quality objectives, standards, and performance metrics aligned with organizational goals.
- **Empower Employees** - Empower employees to take ownership of quality by involving them in decision-making processes, problem-solving initiatives, and continuous improvement

efforts. Encourage open communication and collaboration to foster a culture of accountability and innovation.

- **Address Resistance** - Anticipate and address resistance to change by engaging with employees, addressing concerns, and providing support and guidance throughout the implementation process. Encourage feedback and create opportunities for dialogue to address any misconceptions or apprehensions.
- **Celebrate Successes** - Recognize and celebrate achievements and milestones in quality mindset implementation to reinforce positive behaviors and foster a culture of continuous improvement. This can include acknowledging individual and team contributions, sharing success stories, and rewarding exemplary performance.
- **Monitor and Adjust** - Continuously monitor progress towards achieving quality mindset goals and be prepared to adjust strategies and approaches as needed. Solicit feedback from employees and stakeholders and use data and performance metrics to evaluate the effectiveness of initiatives and identify areas for improvement.

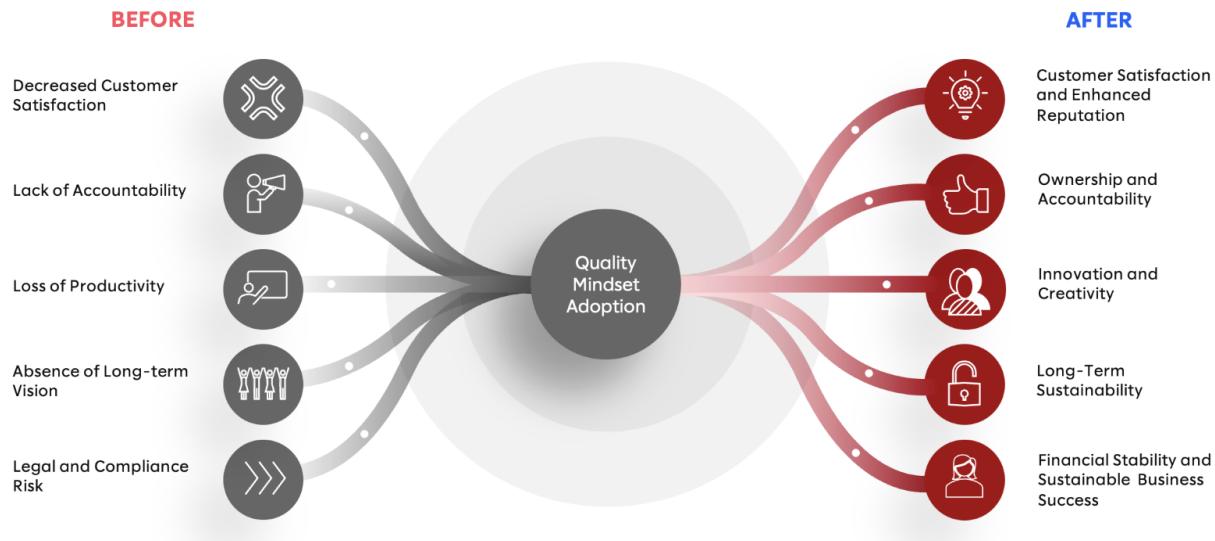
Effectively addressing Quality Mindset implementation challenges requires a deliberate and strategic approach focused on overcoming obstacles and nurturing a culture of quality throughout the organization. By applying targeted strategies and initiatives, organizations can successfully navigate the complexities of quality mindset implementation and drive positive transformation across all levels of the organization. Through proactive measures such as identifying specific challenges, creating awareness, providing comprehensive training, setting clear expectations, empowering employees, addressing resistance to change, and celebrating successes, organizations can cultivate an environment where quality is valued and embraced. By prioritizing these efforts and fostering a collective commitment to quality excellence, organizations can not only overcome implementation challenges but also instigate lasting positive change that benefits both the organization and its stakeholders.

## 6. Conclusion

Fostering a quality mindset is an ongoing process that demands commitment and engagement from all organizational levels. It's crucial to continually reassess and adapt strategies based on feedback and changing requirements. Adopting a better-quality mindset leads to substantial operational improvements, bolstering customer satisfaction and loyalty through enhanced reliability and consistency, which in turn reduces operational expenses. Furthermore, prioritizing better quality enhances the organization's reputation and competitive edge while boosting employee morale and nurturing a culture of innovation and continuous enhancement, thereby establishing a solid groundwork for organizational success and longevity.

## Quality Transformation: From Before to After

Emphasizing quality as a priority yields **consistency, reliability, and customer satisfaction**, fostering trust and loyalty. On the contrary, neglecting quality results in **inconsistencies, unreliability, and customer dissatisfaction**, eroding trust and leading to missed opportunities.



In a world characterized by uncertainty, complexity, and rapid change, the Quality Mindset serves as a compass, guiding organizations and individuals towards excellence, resilience, and success. By embracing its principles and values, organizations can build a culture of quality that transcends products and services, permeating every aspect of their operations and interactions. As we navigate the challenges and opportunities of the future, cultivating a Quality Mindset will be indispensable in driving innovation, differentiation, and sustainable growth.

### References:

Andersen, E. (2013, May 31). 21 quotes from Henry Ford on business, leadership, and life. Forbes.

<https://www.forbes.com/sites/erikaandersen/2013/05/31/21-quotes-from-henry-ford-on-business-leadership-and-life/>

U.S. Chamber of Commerce. (n.d.). Steve Jobs quotes for business owners. U.S. Chamber of Commerce. <https://www.uschamber.com/co/start/strategy/steve-jobs-quotes-for-business-owners>

# Scaling Technical Work While Maintaining Quality - Leveraging Community Principles

**Katherine Payson**

kath.payson@gmail.com

## Abstract

Despite the promises of AI, the demand for freelance technology professionals is surging, as companies seek niche skills and the traditional profile of full-time employees is being challenged. The global gig economy generated \$204 billion in 2023 and is expected to grow 17% in 2024, with the United States predicted to have more gig workers than non-gig workers by 2027 (Bursztynsky, 2023). Scaling any team while maintaining quality output is an age-old challenge. Many technical organizations have turned to freelance professionals to support their service delivery, but the gig economy model adds a new level of complexity to scaling with quality.

How do you maintain quality as you scale global delivery with freelance gig workers? At large scale, the trust-based model no longer works and, while there are lots of performance metrics and ways to monitor your workforce, this paper challenges you to think a little more creatively.

In my experience, many of these traditional monitoring methods can lead to attrition and resentment if you are not also working on building a connected community within your team. This paper aims to highlight principles from the community management discipline and how they can be applied to scaling a workforce in tandem with traditional performance measures to create a positive, motivated community who in turn organically provides you with a higher quality output.

While this paper speaks often of the gig economy and freelance workers, as that is how my organization has scaled its highly technical team, these ideas and principles can be applied to leveraging community management to scale any team.

## Biography

*Katherine Payson is an accomplished cybersecurity industry operations leader. She is currently the Director of Delivery Operations at Cobalt, the world's leading pentesting as a service provider. In this role, she oversees the execution of over 4,000 pentests annually, delivered by a global community of top freelance security researchers. Katherine is responsible for ensuring the quality and efficiency of Cobalt's pentesting services, helping organizations improve their security posture.*

*Katherine is passionate about leveraging creative tools and approaches to scaling service delivery teams. She is a strong advocate for building a diverse and inclusive workforce and is committed to fostering a positive and motivating work environment.*

*Katherine has over a decade's experience in the cybersecurity space and holds an MBA in Leadership.*

*Copyright Katherine Payson June 2024*

## 1 Introduction

In the '80s, you would never dream of getting in a stranger's car and having them drive you from point A to point B. But here in the 2020s, Uber operates in over 70 countries and executed almost 10 billion rides in 2023. The gig economy, a labor market that relies on part-time positions filled by independent contractors, is booming. In 2024, people tend to associate gig work with app-driven work like Uber or DoorDash, but developers, programmers, graphic designers, and more have turned to side-gigs as a stream of income, largely fueled by the post-pandemic desire to work remotely.

According to McKinsey's 2022 American Opportunity Survey, 36% of employed respondents (roughly 58 million Americans) identified as independent workers. In addition, data from the world bank shows that the global gig economy accounts for up to 12% of the labor market (World Bank Group, 2023). People in the US are electing for these jobs for flexibility and ease of entry while internationally, the gig economy offers promise for women and youth in developing countries (Sharif & Zhenwei, 2023). Digital technology is enabling new solutions and opening unique opportunities for employment matching workers to tasks without many of the cumbersome requirements of hiring a large full time staff. In countries like Bangladesh, Malaysia, and Kosovo, governments are providing online gig work training to specifically support youth, women, and those in the bottom 40 percent of income distribution.

My organization, Cobalt, provides penetration testing at scale. For those not familiar, a penetration test or "pentest" is an authorized, simulated attack by a pentester against a computer system to look for vulnerabilities that could be exploited by a hacker. Every day, my team and I work with clients as they incorporate offensive security into their applications and assets. Offensive security is the practice of actively seeking out vulnerabilities and security weaknesses before a hacker exploits them. In contrast, defensive security teams are responsible for defensive measures, like monitoring, detecting, and responding to security threats. Incorporating both of these approaches is important to maintaining a good security posture.

I joined Cobalt in early 2021 after spending six years with more traditional penetration testing vendors. At a legacy pentesting provider, a customer may sign a full Statement of Work (SoW) with a hefty price tag, only to join an 8-week lead time for a skilled tester. The process is clunky, with kick off calls and requests for resources on both sides, then many months later the customer receives a PDF report that they must then take back to their dev or engineering teams and decipher to create internal tickets for fixes.

Cobalt does it differently: Enter Pentest as a Service, or PtaaS. PtaaS combines manual, human testing with a modern delivery platform to deploy ongoing pentest programs with integrations for ticketing systems, ease of reporting, and no wasted time on procurement. Sounds super-cool right? But how does one build a team that can deliver hundreds of tests at any given time, and start another one at any point, with just 24 hour's notice?

We have a community of about 450 highly skilled penetration testers who work from all around the world. While the gig economy is the basis for our delivery model, scaling still comes with great challenges. As you scale from 4 to 40 to 400 workers, you quickly see that the trust-based model no longer works to ensure quality standards and expectations are met. At Cobalt, managing 4 penetration testers was easy and 1 person could track their deliverables and output. At 40, things got a little tougher, so maybe 3 or 4 project managers could maintain standards, but at 400 we have had to pivot and think dynamically about how to ensure that they excel in their craft and offer our clients the top-tier experience we strive for.

You certainly need the basics in place at this scale. A good vetting system, onboarding, a solid independent contractor agreement, documented guidelines, and a clear code of conduct. However, these

things just lay the groundwork. How can you invest in your growing team members' success so they are driven to invest in yours?

Community management is fundamentally a discipline that focuses on building and maintaining relationships with a group of people who share common interests. Most communities that we think of in the business environment fall into a few categories.

- **Support**, where the goal is to improve customer support and satisfaction, reducing support costs, and allowing members to answer questions for each other.
- **Acquisition**, where the goal is to increase brand awareness, grow market share and drive sales traffic.
- **Contribution**, where the goal is to motivate and accelerate contribution of content
- **Engagement**, where the goal is to increase customer retention.

There are a few others, but because these categories mostly boil down to customer satisfaction and sales. Community Management is typically categorized as a Marketing function within an organization. However, the principles of community management can be applied to a variety of other areas, such as scaling operations, development, and technical teams.

People want a sense of belonging, social connectedness, a group that shares their own values, a sense of identity. I believe technical team leaders can draw a great deal of knowledge from the community management discipline to create these people and purpose connections and, in turn, team members will invest in the organization's success. They will pride themselves on working for you, talk about your community, put in the extra effort and bring other members up with them. Adapting community principles like creating a clear identity, utilizing rewards, actively fostering collaboration, providing clear conflict resolution and cultivating champions are invaluable tools to building a low attrition, high-quality delivery team both in the gig economy and for large internal teams.

## 2 Building the Foundations

More often than not, there is a skill requirement to join a community, particularly in the tech industry. For Uber, it's the ability to drive and work the app competently. For Cobalt, to be considered for a preliminary interview, an applicant must have a minimum of 4 years in the industry with strong, demonstrated abilities in penetration testing. That is then followed by a rigorous vetting process including a technical skills assessment, report writing assessment, and soft skills review.

While we have a duty to find the top talent for our customers, these barriers to entry also create a lot of strength in our community. Making it a bit more challenging to join or reach a certain status in the community creates a stronger sense of social identity and belonging for your members. Effort justification, which stems from Leon Festinger's theory of cognitive dissonance, describes a person's tendency to attribute higher value to an outcome they had to work hard to achieve (Vogl, 2016). If one has to invest more energy and time to gain membership, it can create very strong social identities and highly committed communities. In Cobalt's case, this means heavily vetting applicants, and accepting only 5% of the applications that we receive for our community each year.

Despite the prevalent articles you may see about skilled labor shortages—particularly in the security space—competition can be high with organizations leveraging freelance workers. At Cobalt, we work with top-tier skilled individuals who are sought out by our competitors in the space. Onboarding is the first step to capturing these individuals and gaining their investment in us over our competitors. Onboarding should be an exceptional experience for both parties, it's a crucial opportunity for you to make a first impression,

and to make it more likely that community members will become highly engaged members. Think about what you want them to know, how you want them to feel being

## **3 Developing a sustainable Community**

### **3.1 Clear Identity**

Creating a clear sense of identity for your community is essential. It gives members a feeling of belonging, purpose, and direction—but more importantly, creates boundaries. Members know who is in, why they are in the community, and what they need to do. A well-defined identity helps people feel connected to something bigger than themselves, and builds a sense of camaraderie and shared values. It also makes your community stand out, attracting like-minded people who share your mission and goals. A clear identity also guides decision-making, ensuring that actions and initiatives align with the community's purpose and values.

By building a strong identity, you create a lively and engaged community where members feel valued, respected, and motivated to contribute their time, skills, and expertise. For Cobalt, our community's shared vision is a shared passion for securing our customers assets—and in turn making the internet safer. Our community is in a space where it is imperative that members be lifelong learners and grow as technology changes. We work to align our community, and our organization in general, with the identity of highly skilled learners, working to secure the internet.

### **3.2 Rewards**

A key piece of maintaining engagement is rewarding contribution. Providing members with benefits that must be earned creates growth and avoids attrition. Gamification in community management is essential for scaling a workforce because it provides a structured and engaging way for members to contribute and learn. By incorporating game elements (such as points, badges, and leaderboards) community managers can motivate and retain members while encouraging them to participate actively. Gamification can also encourage a sense of community and belonging, as members work together to achieve common goals and celebrate each other's successes.

However, it is important to implement gamification genuinely and authentically to achieve lasting success. In other words, the rewards and recognition should be meaningful and directly tied to the community's goals and values. Gamification should not be used as a gimmick to manipulate or exploit members. Instead, it should be seen as a tool to enhance the community experience and empower members to make meaningful contributions. By focusing on intrinsic motivation and creating a positive and supportive environment, community managers can create a gamified community that is both engaging and sustainable.

At Cobalt, rewards come in the form of tester levels or promotions, compensation, and visibility for thought leadership from our members and swag. Gamification and reward can be a tricky line, trying to drive quality work from your community through extrinsic rewards can come off falsely. The rewards should be a symbol of proof of a member's investment and status. Swag is incredibly popular in the tech space, where someone must be a proud member of your community to wear your logo out in public at industry events and such, and be excited for others to see that they are a member. While we do have general swag at events (the most popular being a t-shirt with a giant taco on it and the tagline “will hack for tacos”), most of our community swag is tied to milestones with us. For example, “Welcome” swag at 5 tests, a backpack and an achievement coin at 100 tests along other items, and so forth. No, people don't

want another keychain flashlight with your logo on it, but community-specific swag goes a long way in rewarding hard work and dedication at Cobalt.

We tier our contractors at two levels (although we are looking to expand this further). Our most tenured top performers are Leads on our engagements. They lead the testing team for an engagement, and are responsible for ensuring that work is fairly divided, reviewing and validating the team's findings, preparing the final testing report, and providing feedback regarding the test and other members of the engagement team. While we do compensate for these responsibilities, we have found the role on its own creates desire for growth and that motivation has fostered genuine Cobalt champions.

### **3.3 Collaboration**

Collaboration and joint decision-making are imperative to community building, because they cultivate a sense of ownership and belonging among members. When individuals feel like they have a say in the direction and development of the community, they are more likely to be engaged and invested in its success. Joint decision-making also helps ensure that the community's goals and values are aligned with the needs and interests of its members. By working together to make decisions, community members can create a space that is welcoming, inclusive, and supportive of everyone's needs.

While your organization of course has to design and execute a roadmap aligned with business and customer needs, it is imperative that your community also has opportunities to provide input on the organization's direction. This ranges from being encouraged to provide constant and regular feedback on your product and their experience using it, but also visibly involving them in decisions around broader strategy such as eliciting market trend input or competitive analysis feedback.

At Cobalt, the constant feedback loop is quite easily managed through active open Slack channels for bug support and application feedback. For larger initiatives and strategy input we turn to our Pentesterr Advisory Board (PAB), which is a 12-month commitment from a selected group of community members. PAB members share and gain knowledge at monthly community meetings, obtain advanced knowledge of Cobalts plans and direction, and influence the development of our product functionality to meet the community's needs. It's also viewed as an elite group to join, and members can prove effective at the Lead tier I mentioned above. It can be very motivating to be selected to represent the community. While in our function, the members represent our community of contractors, an advisory group like this could also be done with scaled with a community of full-time employees (Although in that case, it is imperative that their day jobs allow them enough room to be effective members of your advisory board without a significant added task to their workload).

Another key note about collaboration is that you should continuously support your community in delivering quality in new ways, as technology and your space change. For example, our testers join us having shown competence in writing reports to our standard, but to support their time and create more consistency we have developed some AI for reporting tools. It offers our testers suggestions on how to improve grammar and tone but also offers input on writing the various report sections based on the findings specific to the test. This is a relatively simple use of AI, but it has gone a long way with our community as we recognize we can streamline and support a task they often find cumbersome; Hackers like to hack, not write. It also helps us further an inclusive community adding significant support for members for whom English is not their first language.

### 3.4 Conflict Resolution

Community management at scale does require effort and resources, and a large part of that work is effective monitoring of violators. In community management, Conflict resolution is crucial to maintaining a positive and respectful environment for all members. When conflicts arise, effective resolution strategies ensure that disagreements are addressed promptly and fairly, preventing them from escalating and damaging the community's overall dynamics. By resolving conflicts constructively, community managers can foster a sense of trust, collaboration, and inclusivity, and this allows members to feel safe and valued, which ultimately contributes to the long-term success and sustainability of the community. If members don't trust each other to follow the rules, their faith in the community and organization can quickly be lost. Just like you working with your peers, where you expect everybody is working towards the same end goals and following certain professional guidelines to do so.

At Cobalt we manage performance on a 3 strikes rule, and sanctions for those strikes are gradual. For example, small violations that may result in a tester not meeting a customer's expectations can result in a strike, but a careless code execution that harms a customer's environment or intentional malicious actions against another member will result in an immediate off-boarding from our community. For individuals, these measures may sometimes seem harsh, but you must enforce your performance expectations for the rest of the community to have faith that both you and they are operating with the same interests.

I learned this one the hard way: For a long time, Cobalt's performance management was built around escalations. So either a customer or peer community member would have to come forward with a concern for a ticket to be opened and investigated, otherwise we functioned largely on a trust model. At the time, we felt this system worked: We were addressing all concerns raised while allowing our community members space to do their jobs. We slowly started to see that this was not actually servicing our community at all. In allowing that freedom, we had actually created a fear to come forward and highlight under performers that were not contributing to our community in line with our expectations. A fear that feedback would result in inaction and that inaction could create tension among peers working together on future engagements.

We developed the strike model along with a more proactive approach to performance management. We can now look at an individual's performance statistically from a variety of data points, as well as anecdotally through anonymous peer reviews. This allows us to initiate performance conversations without peer members being put in a difficult spot. The new policy was very well received by the community, and we actually saw an increase in feedback on things from performance to product updates and beyond. Developing these policies and processes for conflict and performance management puts us all on the same page, and shows our community we are fair and invested in their individual and collective success.

### 3.5 Networking Adoption and Support

Particularly in the tech industry, communities are often subsets of a much larger group and network. As a community owner, we must support our members wider than our own community. For cybersecurity, that involves encouraging penetration testers to attend events and network outside of Cobalt. In-person meet-ups provide an opportunity for our members to connect, collaborate, discuss industry trends, emergent technologies, best practices, and more.

Our community team partners heavily with our marketing team on things like content creation for and by our members, highlighting individuals through our social media accounts and of course events. We sponsor Cobalt community events at DEFCON, the industry's biggest event of the year. DEFCON is a hacker convention held every year in Las Vegas, Nevada with over 30,000 attendees. It spans 3 days with a variety of talks and workshops culminating in lots of vendor sponsored events.. While this is a very high visibility event we also sponsor B-sides events that our members lead and speak at regularly. B-Sides is a global non-profit advancing information security through conferences and events to share knowledge. These are usually smaller, more local events where you can meet industry individuals in your area and learn about border cybersecurity topics. On top of those sponsorships, we encourage Cobalt Core members to reach out when they visit new cities so we can sponsor meetups and meals for Core members to get together in person.

### **3.6 Cultivating Community Champions**

Cultivating community champions within your organization can play a large role in gaining community support on changes. These individuals are passionate about your organization and its goals, and are willing to go the extra mile to contribute to its success. By identifying and nurturing community champions, organizations can leverage their influence and expertise to drive positive change. The previously mentioned Advisory Board is one way to cultivate champions as board members are given visibility and are chosen based on the high quality they provide to the organization. However, at Cobalt we recognize that we also have some organically occurring champions in our community. Individuals who are well known at events outside of our community, individuals that release interesting blogs or have a large social media following along with individuals who are very active in our slack channels and people naturally gravitate towards them as a face of the community. It's important to recognize both avenues champions can be created and recognize who those champions are.

Champions can serve as ambassadors, promoting the community and its initiatives to their networks. They can be effective support in recruiting, onboarding and feedback. However, one of the most valuable things we have driven from our champions is getting them to understand how some less favorable decisions are in the best interest of our business, and a means of ensuring that we can continue to build a thriving community.

For example, as we have scaled we launched pentester logging retention. This is a community-wide change in test requirements that means each tester has to upload their logs to Cobalt after a test ends. As you can imagine, this is a big change for our members and a significant shift away from a heavily trust-based model. However, helping our champions see that this helped us ensure that we continue to meet our enterprise customers' security demands, so that we can continue to grow up-market and bring more work to the community, helped with the greater community understanding on this. Cobalt's champions were able to shift the perspective from what could have been perceived as an invasive ask to a normal part of our process, which allows us to continue to bring them more engaging projects to work on as part of our community.

## **4 Establishing a Community Strategy**

When you give people a genuine sense of community, it motivates them to want to get involved and contribute. Motivate enough people, and a community can help you achieve incredible scale, but how does one build a strategy towards this genuine motivation?

David Spinks who is one of the founders of Bevy, a well-known community platform, presents one of the clearest ways to think about community strategy in his book, *The Business of Belonging*. David outlines the three levels you should consider:

- **Business level** - How your community program will drive revenue for your company. This may differ with your goals. For example, at Cobalt's current stage of growth, this is not directly measured in revenue and the focus is on providing the customer a top-tier quality consistent experience with all of our community members. In turn, we hope this contributes to the customers renewing contracts with us and providing us with positive reviews to peers.
- **Community level** - How your community will grow and become more healthy and engaged over time. For Cobalt, this could be something like decreasing the time to a first post for new members to capture engagement faster or grow attendance on our monthly training and Q&A calls.
- **Tactical level** - The specific initiatives and improvements you work on to build a healthy, engaged community and achieve the business outcomes. At Cobalt, we try to plan this with a quarterly and yearly view, and our actions can include smaller things, like: increasing community content posted on our blog, to larger goals, like introducing a new tiering model to our community to allow more senior resources the opportunity to mentor junior members.

Each of these goals should have metrics attached to track your success. For example, increase community-authored blog posts by x number of articles in the third quarter. The next section outlines a measurement framework you can use to assess the overall health of your community and provide a broader view on how your strategy is working.

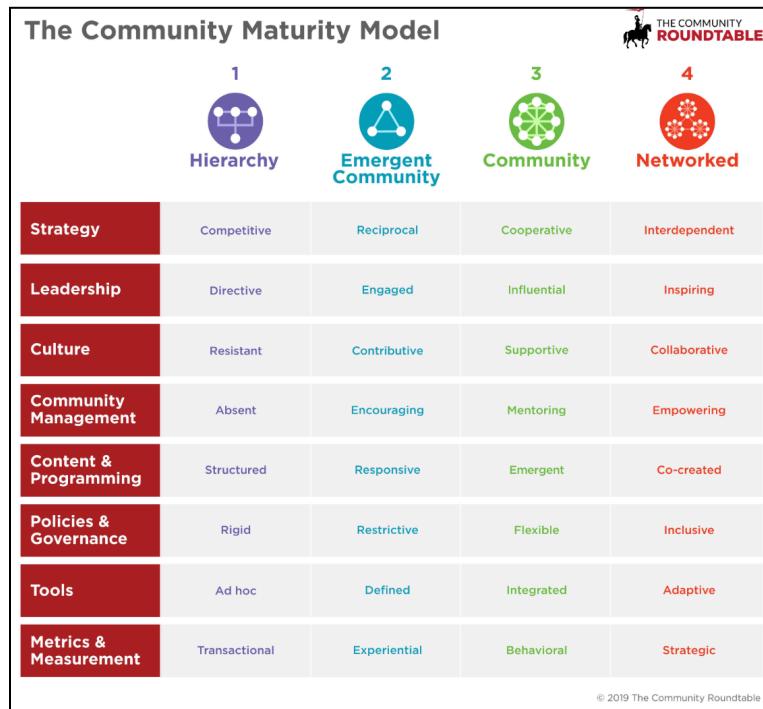
## 5 Measuring Success

There are two fundamental areas you should be measuring the success of your community. The first and perhaps more obvious, is a community member's skill in the applicable discipline and their ability to deliver on their contractual obligations. At Cobalt, measuring a penetration tester's performance looks like a review of their technical skills through vulnerabilities found and attack paths taken, a review of soft skills with customer interactions, and more tangible numbers like a peer review score or customer Net Promoter Score (NPS). For those not familiar, an NPS score is a market research metric that is based on a single survey question asking respondents to rate the likelihood that they would recommend a company, product, or a service to a friend or colleague. These areas are imperative and fundamental to ensuring you are delivering for your customers, but the second area you should be measuring is whether your community efforts are having impact and resulting in a more quality output?

The first step is to put a framework around your community maturity and lay a foundation for your community strategy. Every organization with a community function should have a maturity model as a way to measure their success. A maturity model will serve as a shared expectation for how your community develops, and provide a framework that allows benchmarking for the current state and identifying areas of improvement for your community practices. There are a few published frameworks for community management, but my favorite is from The Community Roundtable, previewed below. You can customize these frameworks to your organization, and weigh some areas more or less, depending on your goals. Either way, a framework allows you to perform a gap analysis and see where your efforts may be best spent.

The diagram below shows 8 competencies on the left that you can measure for your community. These are broad but could be customized for your specific community. Then the ratings from left to right articulate how these competencies progress from hierarchical organizations to those that have embraced

a networked approach to their business. Hierarchical being an ad hoc use of some of these areas to networked which would imply a full corporate strategy behind your community.



Building a community is not a linear timeline, but rather a series of these maturity indicators or milestones that lead to a more networked organization, in this example.

For the next step, you have to ask your community about how satisfied they are with your strategy and their day-to-day life in your community. A strong indicator of this can be a simple NPS score from the group, but you can also survey around specific events or initiatives. We survey our community every other month, which provides us with a simple NPS score and allows us to ask dynamic questions based on our business needs and about our community's interests and needs. For example, we can ask how many members feel they are skilled at a new service we are looking at offering, or what kind of events our testers are planning on attending in the next 6 months and how we can support them.

And the third step is gathering some very simple measures as you're getting started, such as engagement. Measuring engagement within a community allows you to get a gauge on the level of involvement and participation of your members. This can be done by tracking various metrics, such as the number of active participants, frequency of interactions, contributions to discussions, content creation, and event attendance. By analyzing these simple metrics, you can assess the success of your efforts and also identify areas for improvement. The sooner you can start collecting this data, if you are not already, the sooner you can build a baseline to compare to your future metrics and see the value of your community strategy.

## 6 Conclusion

Large technical teams can significantly benefit from adopting community management principles to scale their operations and drive innovation. By fostering a sense of community among team members, organizations can create an environment where individuals feel valued, connected, and motivated to contribute their best work.

Building a successful community requires a comprehensive approach that incorporates aspects, such as effective communication, conflict resolution, community adoption and cultivating champions. By implementing these strategies, organizations can create thriving communities that drive innovation, foster collaboration, and contribute to the overall success of the organization.

Continuously evaluating and refining your community management strategy and practice is essential to adapting to the evolving needs of members, and to aligning with the organization's strategic goals. By investing in community building, organizations can unlock the potential of their members, create a sense of belonging, and position themselves as leaders in their respective industries.

## Disclosure & Acknowledgement

Gemini for Google Workspace was used to proofread this document. Gemini proofread includes checks for spelling, grammar, active voice, sentence splits and conciseness.

## References

- Bacon, Jono, and Peter H. Diamandis. *People powered: How communities can supercharge your business, brand and teams*. Nashville, TN: HarperCollins Leadership, 2024.
- Bursztynsky, Jessica. "What's in Store for the Gig Economy in 2024, According to 7 ..." Fast Company, December 22, 2023.  
<https://www.fastcompany.com/90999951/whats-in-store-for-gig-economy-in-2024>.
- "Community Maturity ModelTM." The Community Roundtable, March 5, 2024.  
<https://communityroundtable.com/what-we-do/research/community-maturity-model/>.
- Delouya, Samantha. "The Rise of Gig Workers Is Changing the Face of the US Economy | CNN Business." CNN, July 25, 2023.  
<https://www.cnn.com/2023/07/24/economy/gig-workers-economy-impact-explained/index.html>.
- Mancini, Maggie. "The Rise of Gig Work in 2023." HRO Today, March 28, 2023.  
<https://www.hrotoday.com/news/hr-news-north-america/the-rise-of-gig-work-in-2023/>.
- Sharif, I. S., & Zhenwei Qiang, C. (2023, September 7). *The promise and peril of online gig work in developing countries*. World Bank Blogs.  
<https://blogs.worldbank.org/en/voices/promise-and-peril-online-gig-work-developing-countries>
- Spinks, David. *The business of belonging: How to make community your competitive advantage*. Hoboken, NJ: John Wiley & Sons, Inc, 2021.
- Vogl, Charles H. *The Art of Community: Seven principles for belonging*. Oakland, CA: Berrett-Koehler Publishers, a BK Currents Book, 2016.
- World Bank Group. "Demand for Online Gig Work Rapidly Rising in Developing Countries." World Bank, September 7, 2023.  
<https://www.worldbank.org/en/news/press-release/2023/09/07/demand-for-online-gig-work-rapidly-rising-in-developing-countries>.

# Fusing Shift Left Testing, AI Solutions & Automation to Elevating Testing Efficiency

Peh Wei Wooi, Ang Kok Cheang

([wei.wooi.peh@intel.com](mailto:wei.wooi.peh@intel.com), [kok.cheang.ang@intel.com](mailto:kok.cheang.ang@intel.com))

## Abstract

Catching defects early in the software development lifecycle (SDLC) is essential for achieving a cost-effective, timely, and high-quality software release, ultimately mitigating project risk. The shift left approach advocates advancing testing toward the early phases, emphasizing enhancing software quality, and minimizing critical defects in later stages. This strategy aims to streamline efforts and reduce costs associated with defect fixing. Our solution is to infuse the AI (Artificial Intelligence) & Automation solutions at the earlier stage of SDLC. In this dynamic and insightful presentation, we will embark on a journey to explore the transformative capabilities of chatbots, conversational AI, and Postbot in the ever-evolving digital landscape. LC to enhance work efficiency, improve software quality, and accelerate time-to-market with minimal cost.

**Requirement** – AI to analyze large datasets of user feedback, customer/market requirements, and historical project data to provide insights that help create more accurate technical specifications.

**Design** – AI to analyze and generate design prototypes (Optimized & best practices) based on the technical specifications to accelerate the design phase. AI-based tools can automate the generation of code snippets or templates based on design specifications, reducing the manual effort required for coding repetitively. AI-powered automation tools can automate various testing activities such as unit testing, integration testing, functional testing, or performance testing. This ensures early detection of defects and validation of design decisions, leading to higher software reliability.

**Development** – AI to provide code analysis, code snippets generation, potential defects detection, and security vulnerabilities. Automation frameworks to enable continuous integration & development (CI/CD), automating the software build, enabling unit & smoke testing (Shift-left Approach), etc. AI also helps to analyze data to estimate task duration & allocate resources efficiently.

**Testing** – AI to generate the test cases based on the technical specifications, prioritize the test execution based on the software risk analysis, and predict potential defects based on code changes & historical data. Automated test cases such as functional testing, integration testing, performance testing, etc to ensure comprehensive test coverage and early defect detection.

In this paper, we will elaborate on our solution for revolutionizing the software development lifecycle (SDLC) by integrating AI and automation at its core, emphasizing the principles of the shift left approach. By infusing AI and automation solutions early in the SDLC, we strive to incubate high efficiency, elevate software quality, and expedite the development and testing processes, ultimately accelerating time-to-market.

## Biography

Peh Wei Wooi is the Platform Validation Manager in the Intel Platform Integration & Validation (PID) organization. His role and responsibilities are to develop and maintain the Intel platform validation process, methodologies, test plans, and BKM's for technologies and products. Before this role, he was a Software ADP (Application Development Program) lead in Motorola Solutions who was responsible for supporting all the API-related inquiries, proof of concept/Demo, system integration, etc. from ADP partners globally. He has also been certified as the ISTQB (International Software Test Qualification Board) tester.

Ang Kok Cheang is the Platform Validation Architect in the Intel Platform Integration & Validation (PID) organization. His role and responsibilities include defining platform validation strategies, overseeing comprehensive testing, conducting risk assessments, and maintaining detailed documentation for Intel Client products.

# 1 Introduction

According to the NIST (National Institute of Standards & Technology), resolving defects in production can cost 30 times more if compared to other SDLC (Software Development Lifecycle) pipelines.

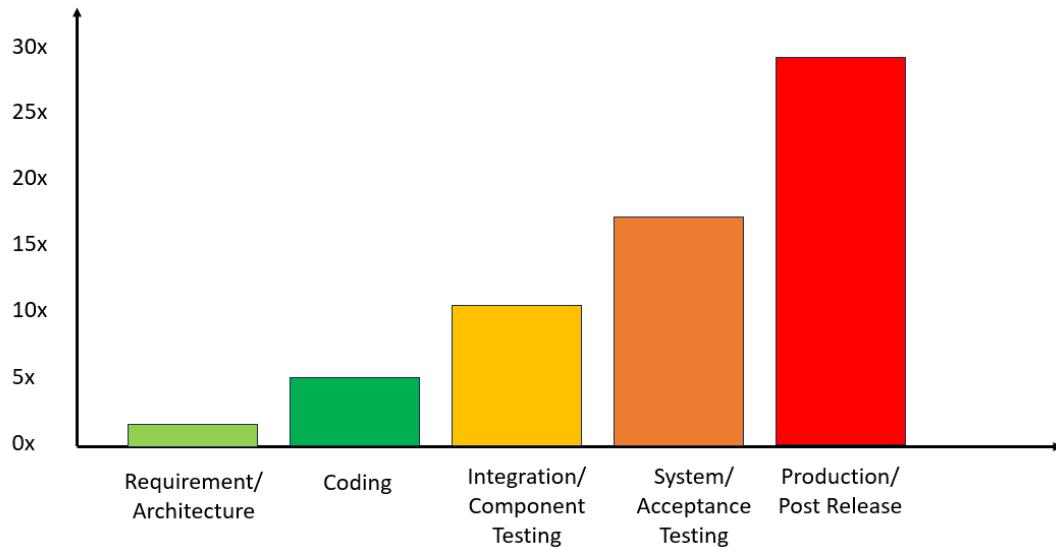
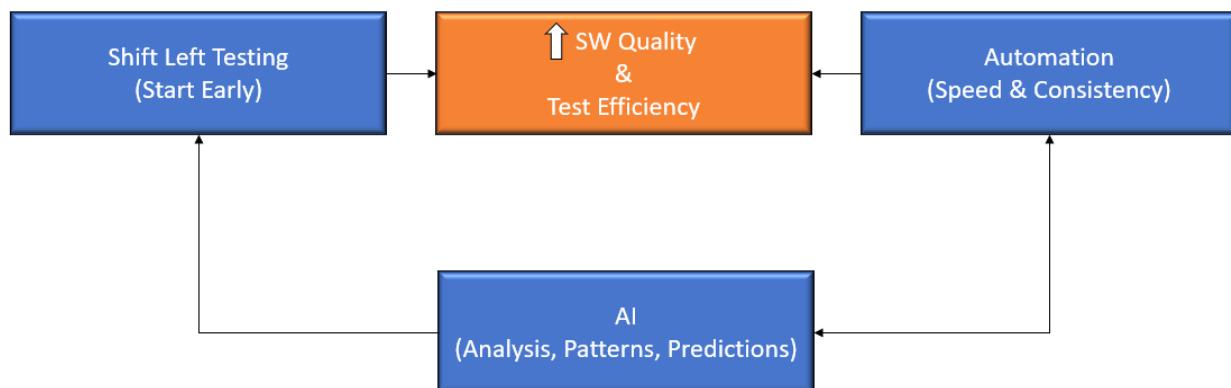


Figure 1: Relative Cost to Fix Defects vs Time of Detection

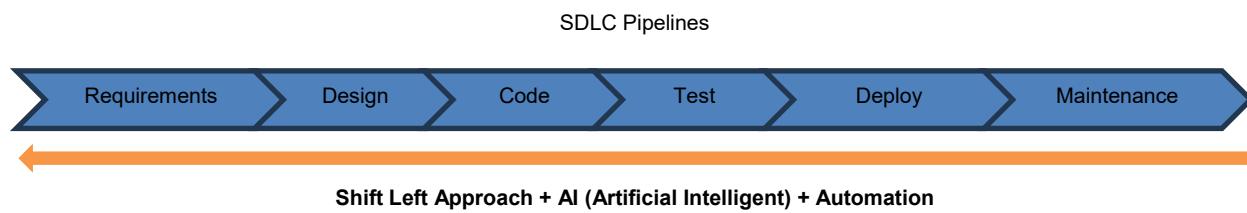
Shift Left Testing approach intends to identify defects as early as possible in the software development lifecycle, resulting in:

- a) Reduced cost, effort & time spent on rectifying the defects.
- b) Increased efficiency in the software development lifecycle.
- c) Increase product quality.
- d) Increasing time to market.

We aim to develop a transformative testing solution that seamlessly facilitates the adoption of the shift left testing approach, leveraging the integration of Artificial Intelligence (AI) & Automation capabilities to achieve a more proactive, efficient, and high-quality software development process.



## 2 Elevating SDLC with Synergy of Shift Left, AI & Automation



Shift Left Testing is a practice in software development where testing is performed earlier in the Software Development Life Cycle (SDLC) rather than at the end. The phrase "shift left" refers to moving testing activities to the "left" side of the project timeline, typically during the requirements, design, and coding phases. The benefits of shift left testing will be early defect detection, faster time to market, improved software quality, reduced costs, better alignment with requirements, etc.

### 2.1 Requirement Phase

The Requirement Phase is crucial as it lays the foundation for the entire development process. The goal is to gather, analyze, document, and validate the requirements of the software product to ensure that the final outcome meets the stakeholders' needs. Artificial Intelligence (AI) can help to analyze large sets of requirement documents (From customers, markets, legacy products, etc) to identify patterns, inconsistencies, and gaps. Natural Language Processing (NLP) algorithms can be used to ensure the requirements are defined clearly and unambiguously. AI can also help to predict the potential risks based on historical data to help enable proactive mitigation strategies. Meanwhile, AI can automate requirement generation based on predefined templates and past projects to ensure consistency & save time/effort.

**Requirement Gathering** - NLP tools can help to analyze and extract the requirements from unstructured data sources (Such as meeting transcripts, emails, user stories, documents, etc). These tools can identify key requirements, stakeholders' needs, or potential ambiguities via these large volumes of text. AI algorithms can also group similar requirements with the same content to eliminate redundancy. Automation tools, it can help to fasten the requirements collection and consolidation progress to improve efficiency.

**Requirement Analysis** - AI can help to provide sentiment and pattern analysis. Sentiment analysis will be based on the stakeholder's inputs to evaluate the urgency of the needs. Based on the output, we can further clarify with stakeholders which are the critical needs for the product. For pattern analysis, AI can help to identify the trend of past programs and propose requirements that might have been overlooked or predict potential defects with the current set of requirements and automation can continuously analyze large datasets with any human intervention. AI-driven NLP models can also parse textual requirements and extract key entities, actions, and relationships. Automation tools can trigger automated follow-up actions or changes to clarify high-priority requirements with stakeholders to ensure critical needs are addressed promptly and accurately.

**Requirement Categorization & Prioritization** - AI models can classify the requirements into different categories such as functional, non-functional, Security, Performance, etc. This classification is based on the predefined rules and then ranks the severity of requirements based on factors like stakeholders' importance, potential impact, technical complexity, etc. Automation tools can generate the requirements using predefined templates ensuring uniformity and minimizing the time spent on manual documentation.

**Requirement Validation & Verification** – AI-driven automation tools use rule-based engines and machine learning models to automatically check for inconsistencies (Such as terminology, phrasing, or intent) or ambiguities (Detect the ambiguous phrases that might be open to interpretation) within the requirements via ontological analysis & cross-referencing to ensure the requirements are defined clearly, consistent with industry standards and without any contradiction. By automating this validation, precision in requirements is significantly enhanced and ensures the product is meeting all necessary standards.

**Requirement Prediction & Estimation** - AI can forecast the effort, cost, and time that is required to implement the specific requirements based on historical data and provide the impact analysis to the user. Automation tools continuously gather and analyze the data from sources and feed them back to AI models. When a requirement is modified, AI with automation capability can help to assess the impact of the change across the project in real-time continuously to make sure the modifications are carefully handled. With this, the AI can provide the risk assessment according to the timelines, budgets, etc to the project for decision-making.

The key strategies for implementing Shift Left in the Requirement Phase will be early validation and verification of the requirements to ensure the requirements are accurate, complete, and aligned with customer needs. Automated the requirement analysis by utilizing the NLP tools and AI-driven techniques to automatically analyze requirements for ambiguities, inconsistencies or missing details. Adopt Behavior-Driven Development (BDD) software development approach that focuses on collaboration between developers, testers, and business stakeholders to gather early feedback and create a shared understanding of the deliverables, while Test-Driven Development (TDD) helps to ensure that the code meets the desired requirements from the start, and improves the overall design, quality, and maintainability of the software.

## 2.2 Design Phase

The design phase is crucial as it transitions from the "what" of the requirements phase to the "how" of building the software. This phase focuses on creating a blueprint or architecture for the software system, ensuring that all requirements are met in a structured and efficient manner. Incorporating early and continuous testing, promoting collaboration, and implementing continuous feedback loops in the design phase ensures early detection of design flaws, alignment with requirements, and ongoing refinement based on user and stakeholder input.

**System Design Phase** - Involved in both High-level design (Architecture, Technology stack, Interfaces, Data design, Security design, etc) and Low-level design (Component design, Algorithm design, User Interface design, Error handling & logging, etc). AI can analyze requirements and automatically propose architectural patterns or frameworks that best fit the project's needs. For example, AI can propose microservices architecture for highly scalable or a monolithic architecture for simpler projects. AI algorithms can optimize the design by evaluating various architectural options against criteria such as performance, scalability, and budget. This allows for selecting the most efficient architecture early in the design phase. AI can automate the creation of data models by analyzing the requirements and generating the relationships and data flow diagrams. AI-driven automation tools also can automatically generate designs and API specifications based on functional requirements to ensure they align with the overall system architecture. AI-driven automation tools can perform threat modeling, identifying potential security vulnerabilities in the design and suggesting mitigation strategies accordingly and continuously monitor the design against industry standards to ensure the security measures are incorporated correctly.

**Prototyping** - AI tools can generate UI/UX mockups based on the requirements, providing visual representations of the interface without manual design work. These mockups can be generated automatically and quickly iterated upon based on stakeholder feedback. AI can predict user behavior and optimize the user interface design for better usability that includes suggesting layout changes, navigation improvements, accessibility enhancements, etc. AI can perform feasibility testing via automation to test various design concepts (Proof of Concept) through stimulation before full-scale development.

**Design Review** - AI can help perform design validation in real-time via automation to identify inconsistencies, missing elements, or potential flaws. AI can facilitate peer reviews by automatically comparing the design against best practices, and coding standards then highlighting the areas where the design deviates from established norms.

**Design Documentation** - AI tools can generate design diagrams such as flowcharts, Entity Relationship Diagram (ERD), etc from the design specifications automatically. AI-powered NLP can help in drafting clear and concise design documentation by summarizing technical details and creating readable content based on the design data. AI can also automatically generate the traceability matrix that links design elements to the respective requirements. This ensures that every requirement is addressed and aligned with the project goals.

**Preparation of Implementation** - AI can break down the design into detailed implementation tasks and assign them to the team members accordingly based on their skills to optimize resource utilization. AI can automate the setup of the development environment such as version control setup (Repository initialization, Branch & merging automation, etc), build server configuration (CI/CD pipeline creation, dependency management, etc) and environment dependency setup (Infrastructure as Code, Containerization/Virtualization, etc) to ensure everything is ready for implementation.

**Risk Assessment** - AI can predict potential risks in the design phase by analyzing historical project data and current design choices. This allows for proactive risk management where potential issues are identified and mitigated before they become problematic. AI can suggest mitigation strategies for identified risks, such as alternative design approaches to safeguard the project from unexpected challenges.

The key strategies for implementing Shift Left in the Design Phase will be using automated AI-driven tools to analyze requirements and suggest the most appropriate architectural patterns, frameworks, or technology stacks based on specific criteria such as performance, scalability, and cost. Fostering a continuous feedback and collaborative environment to allow real-time feedback on design decisions and enabling quick iterations and refinements. Integrate AI tools for threat modeling during the design phase to identify potential security vulnerabilities and suggest mitigation strategies proactively. AI can continuously monitor the design against industry standards to ensure all security measures are correctly incorporated, recommending necessary real-time adjustments to maintain a secure design posture if the design changes.

## 2.3 Code Phase

The coding phase, also known as the implementation or development phase, is where the actual creation of the software takes place. This phase involves translating the design documentation into executable code that meets the defined requirements. Involvement in code testing at the earliest stages helps catch and fix bugs before they become more costly and complex. Meanwhile, encourage collaboration between software developers and testers to ensure improvement of code quality.

**Code Development** - AI-powered code editors use machine learning models trained on large codebases to suggest code snippets, functions, or entire blocks of code. This speeds up the coding process by reducing the need for manual coding of repetitive patterns. Automation tools can help to generate boilerplate code based on predefined templates or project-specific configurations. AI-driven tools can analyze code as it is being written, identifying syntax errors, potential bugs and performance issues in real time that allows developers to correct the mistakes immediately. AI can also suggest refactoring opportunities to improve the code quality by simplifying the complex logic, removing redundant code, or converting inefficient loops. Automated refactoring tools can apply these changes with minimal developer intervention.

**Code Reviews** - AI can assist in code reviews by automatically analyzing code for adherence to coding standards, detecting potential security vulnerabilities, and identifying code smells. AI can learn from the previous code reviews and continuously improve its ability to detect issues such as highlight patterns or mistakes frequently made to avoid similar mistakes from happening again.

**Unit Testing** - AI can analyze the code and automatically generate unit tests that cover various execution paths, edge cases, and potential failure points. This ensures comprehensive test coverage without requiring developers to manually write the test cases. AI can also help to prioritize which tests to run based on the likelihood of code changes affecting certain parts of the system. This will smartly reduce the time spent running unnecessary tests and focuses resources on the most critical areas only.

**Integration of Components** - AI able to predict and automatically resolve merge conflicts by analyzing the changes made in different branches. This reduces the manual efforts by automated merging conflict resolution into the Continuous Integration (CI). AI also optimizes the CI process by analyzing past build performance, and dynamically adjusting build configurations (E.g. parallelizing builds, allocation of resources) to minimize build time and resource usage. When updating the dependencies, AI can automatically resolve conflicts between different versions of libraries by applying the best possible combination of dependencies through automated dependency management.

**Debugging & Optimization** - AI can detect potential issues by analyzing the code patterns, historical data, and common coding mistakes. AI can also predict potential issues based on the current state of the codebase, warning developers of areas that are likely to fail during runtime. This proactive approach helps prevent bugs before they occur. With regards to code performance optimization, AI-driven profiling tools can analyze code performance in real-time, identifying bottlenecks and suggesting optimizations. For example, AI may suggest more efficient algorithms or data structures based on the specific context of the applications.

**Documentation** - AI can provide intelligent documentation management by tracking changes in the codebase and automatically update related design documents, ensuring consistency between the documentation and the actual implementation. This reduces the manual effort required to maintain accurate documentation.

**Final Integration & Build** - AI can assist in analyzing and optimizing the final process such as minimizing build artifacts, reducing the size of deployable packages, or optimizing the order of build steps to save time. AI can also automate the preparation of the final build for deployment, including setting up configuration files, generating release notes, and creating the deployment scripts.

The key strategies for implementing Shift Left in the Code Phase will be early and continuous code analysis via AI tools to obtain real-time feedback on syntax errors, potential bugs, and performance issues. Utilize static code analysis tools to automatically check code against predefined standards, security vulnerabilities, and code smells during development. This enables developers to fix issues as they code rather than waiting for later phases. Automate the code reviews to automatically analyze the code for adherence to coding standards, security vulnerabilities, and common coding errors and identify patterns or common mistakes to avoid repeating them. Automate unit testing to prioritize critical test cases and enable early detection of defects.

## 2.4 Test Phase

The Testing Phase is a comprehensive process that ensures the product is robust, functional, secure, and meets all specified requirements. It involves planning and designing tests, executing those tests, identifying and resolving defects, and validating the software's performance, security, and usability. The goal is to deliver a high-quality product that is ready for deployment and meets user expectations.

**Test Planning** - AI can analyze the requirements, historical data, and risk factors to automatically prioritize the test cases based on their importance. Any impact on the code change or bug fix will also be taken into consideration.

**Test Case Design** - AI can automatically generate test cases by analyzing the software's requirements and design documents. NLP models can extract key entities, actions, and conditions from the textual requirements and convert them into structured test cases. AI algorithms can also help to optimize test cases that are overlapping or eliminating unnecessary tests to ensure maximum coverage with minimum effort.

**Test Environment Setup** - Automation tools can be used to configure the test environment such as deploying software, setting up the database, configuring network settings, etc. AI can help in monitoring the resource usage of the test environment, scaling the resource based on demand to ensure no over or under-utilizing of resources.

**Test Execution** - Automated testing tools can be integrated into the CI/CD pipeline, allowing tests to be executed automatically when new code is released. Automation allows tests to be run in parallel across multiple environments or configurations to shorten the test cycle duration. AI can also help to dynamically analyze and optimize the test suite based on real-time changes and automatically update the test scripts to adapt to these changes to reduce maintenance burden.

**Defect Reporting & Tracking** - AI models can predict potential defects by analyzing code changes, historical data, etc. Automation tools can automatically log defects detected during test execution, complete with detailed information about the conditions under which the defect occurred. In terms of defect triage, AI can classify and prioritize defects based on their severity, impact, and likelihood of occurrence. AI can assist in identifying the root cause of defects by analyzing the error logs captured via Automation tools or provide insights that help developers address underlying issues more efficiently.

**Test Automation** - AI can automatically generate test scripts for various types of testing such as functional, regression, performance, etc by analyzing the test case's procedure. AI can also monitor changes of the test procedure and update the test scripts automatically to reflect those changes accordingly.

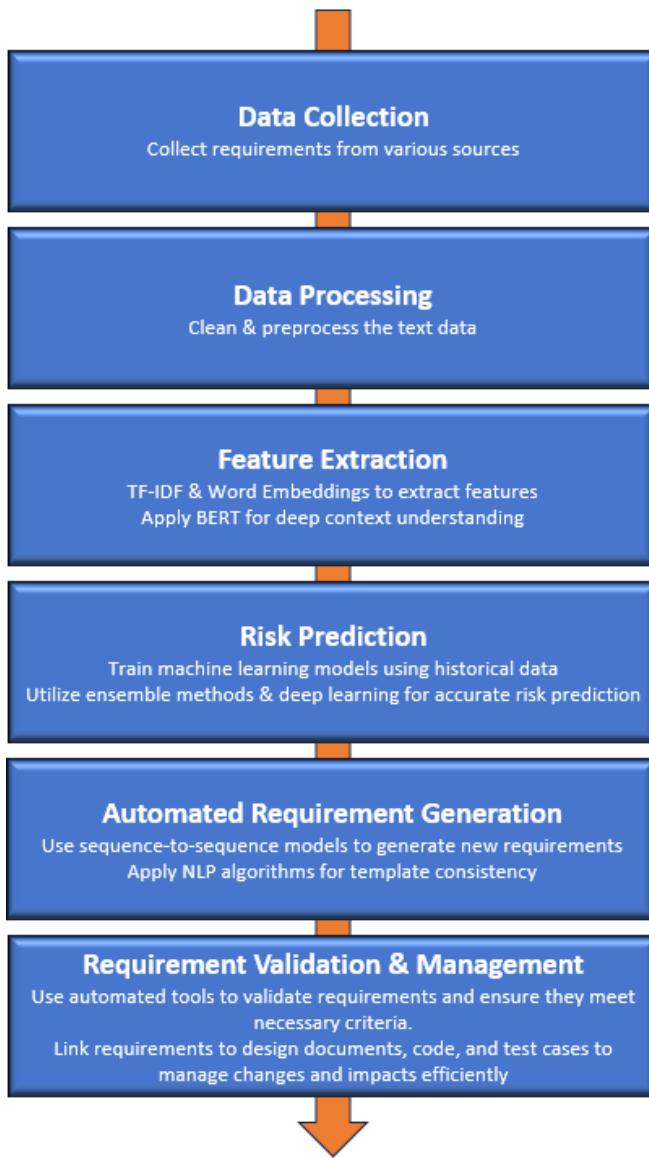
**Test Reporting** - Automation tools can generate the detailed test report including pass/fail status, defect summaries, and test coverage metrics with predefined template. AI can enhance these reports by providing insights and recommendations based on the data and creating a dynamic real-time dashboard that displays the current status of testing, including test execution progress, defect status, and risk indicators. These dashboards provide stakeholders with immediate insights for faster decision-making.

The key strategies for implementing Shift Left in the Testing Phase will be to incorporate testing early by engaging testers during requirements gathering and start the test planning process as soon as the requirements are gathered. This allows the early risk-based test planning enablement to assess (via AI tools) which areas of the software are most likely to have defects. Involvement of early automation strategy by automating as many tests as possible such as unit tests, API tests, UI tests, etc from the start of the program. This will ensure that any new code or change is tested immediately. Integrate automated tests into the Continuous Integration/ Continuous Development (CI/CD) pipeline. Every commit triggers automated tests to ensure that defects are detected early in the development cycle. Early test case design from requirements by automatically analyzing requirements and design documents. Shift unit testing to software developers by ensuring that the software developers write and execute unit tests for the code immediately. Conduct component testing in parallel with development. The automation framework enables independent testing of individual components before integration into the larger system. Additionally, performance tests (such as stress & stability, KPI benchmarking, etc.) can begin concurrently once specific component tests have successfully validated their functionality.

### 3 Integrating New Strategy within Workflow

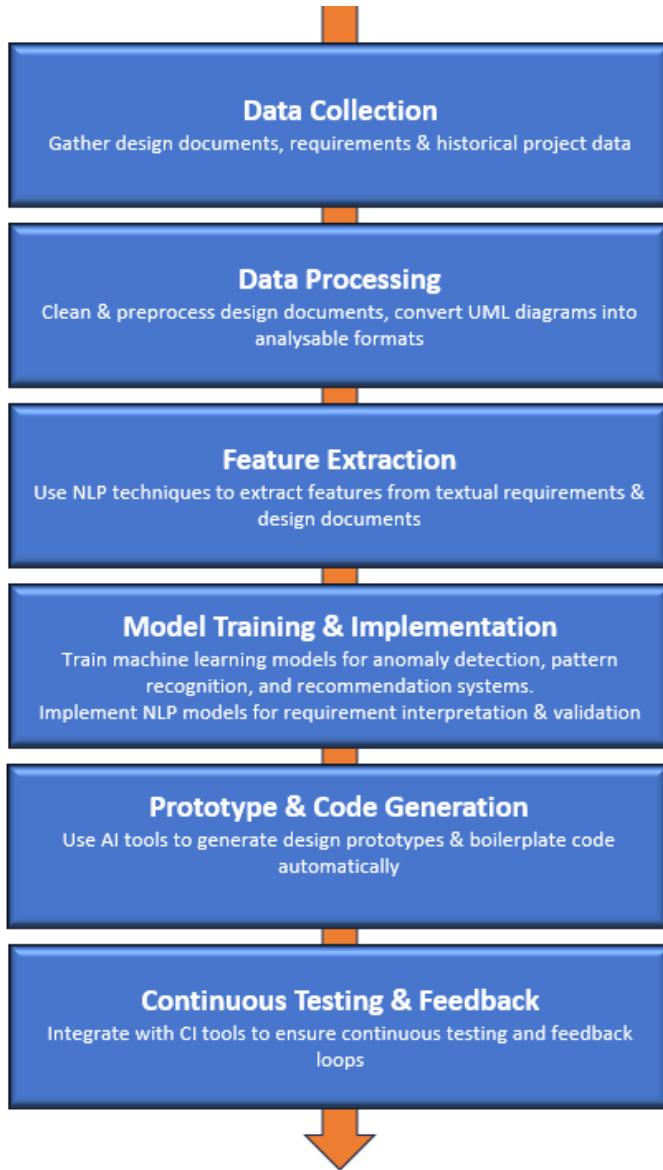
#### Requirement Phase

A comprehensive AI methodology that integrates multiple techniques can be outlined. The methodology should leverage Natural Language Processing (NLP) for text analysis and machine learning for risk prediction, alongside automation techniques for requirement generation.



## Design Phase

Ensures early detection of design flaws, alignment with requirements, improved requirement gathering and interpretation, and consistent, high-quality design outputs. Design Pattern Recognition & Similarity Analysis can be used for automated design recommendations. In terms of predicting potential issues, the approach may involve Anomaly Detection or Pattern Analysis methodologies.



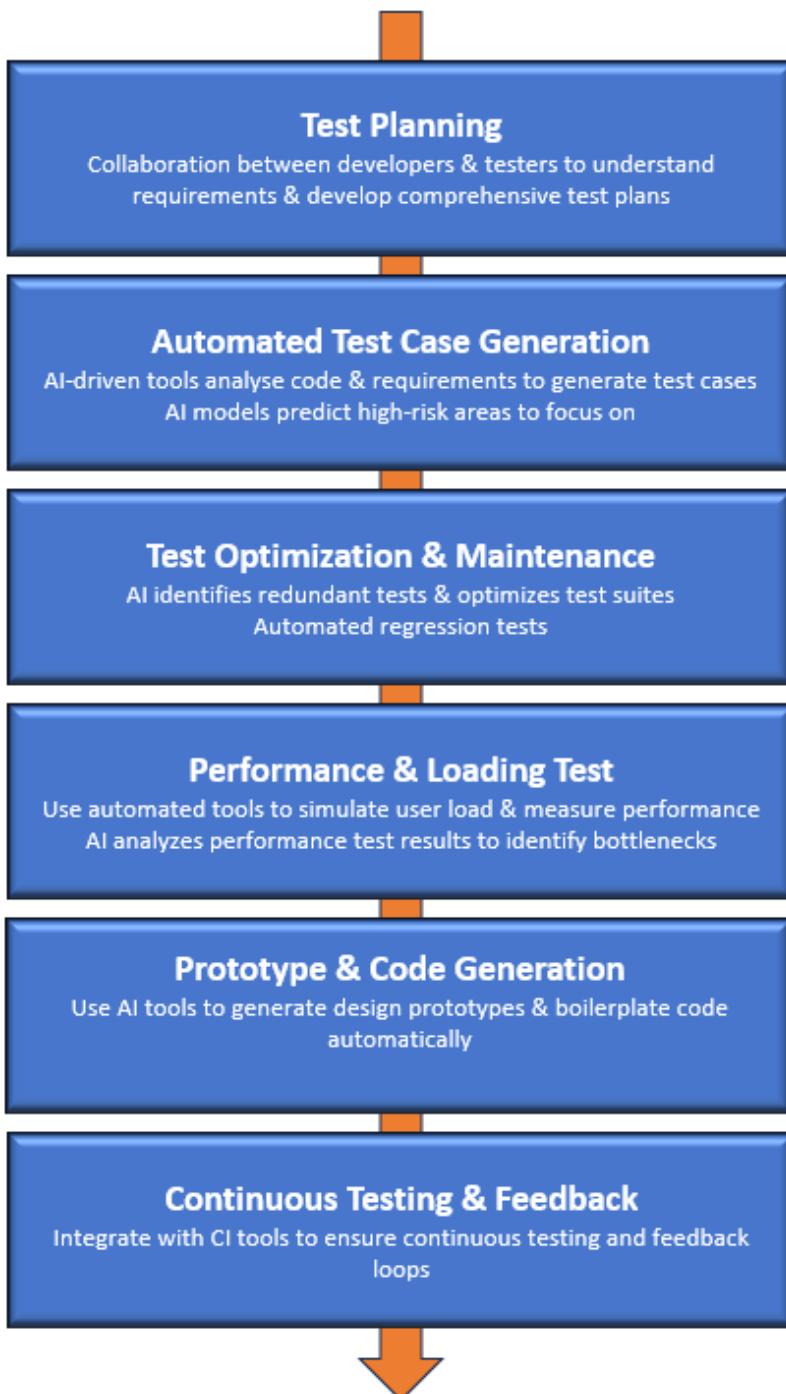
## Code Phase

Ensures early detection and fixing of bugs, promotes collaboration and maintains a clean and maintainable codebase. AI tools can be used to automatically review code for potential issues, and code smells to ensure adherence to coding standards. Predictive Analytics or Pattern recognition can be used for bug prediction and prevention.



## Testing Phase

Ensures comprehensive test coverage, identifies high-risk areas, optimizes the test suite, and maintains software stability and performance, ultimately leading to more efficient and high-quality software. Use AI tools to analyze code, requirements, and historical data to generate comprehensive test cases. Natural Language Processing (NLP) and machine learning algorithms can interpret requirements and code to create relevant test scenarios. Defect Pattern Analysis & Risk-Based Testing can be used for failure prediction. AI can be used to identify redundant test coverage to optimize & streamline the testing process.



## 4 Challenges in Implementation

Data	Integration	Talent	Culture	Cost	Security

### **Data Availability & Quality**

Data needed for training the AI models may be dispersed across different systems and formats, making it difficult to gather and unify. Meanwhile, AI models require large amounts of high-quality data to train effectively, which may not always be available.

### **Integration Complexity**

Integrating different tools such as AI, Automation, and Testing into a cohesive workflow can be challenging as it requires careful planning and significant effort for customization to ensure seamless operation.

### **Talent & Skill Gaps**

The team may require extensive training to gain the knowledge to use and maintain AI and automation tools effectively. There is a need for skilled professionals who understand testing processes and AI or machine learning techniques.

### **Cultural and Organizational Resistance**

Shift-left requires a cultural change within the organization and there can be resistance from teams accustomed to the legacy testing methods. Collaboration between software developers, testers, and AI/Automation specialists is necessary but difficult to achieve due to a lack of comprehensive knowledge, communication barriers (Different technical jargon, distributed team, etc), and organization silos.

### **Cost & Resource Allocation**

Implementing AI and automation requires a substantial initial investment in tools, infrastructure, and training. Skilled resources are essential for developing, implementing, monitoring, maintaining, and updating AI models and automation scripts.

### **Security & Compliance**

All data used in AI and automation processes must be secured (Due to sensitive data, proprietary, etc) and comply with relevant regulations, including adherence to industry standards.

## 5 Overcoming Challenges

### **Data Availability & Quality**

Start by clearly mapping the data sources and defining data governance practices. Ensure that the data pipeline is well-documented and understood by all stakeholders. Invest in Extract, Transform, Load (ETL) tools that can clean, transform, and unify data from various sources into a centralized repository. AI-driven tools can automate data cleaning and normalization. Establish a data quality framework to continuously monitor and assess data integrity, completeness, and accuracy. Train the team with a growth mindset in data management and the importance of quality data in AI. Promote a culture where learning from mistakes related to data handling is seen as an opportunity for growth.

### **Integration Complexity**

Break down the integration process into smaller, manageable modules. Focus on integrating one tool at a time and gradually expanding the workflow. Utilize middleware solutions that can facilitate communication between different tools. Middleware can help standardize data formats and protocols across tools, simplifying integration. Encourage an API-first approach to ensure that all tools and platforms can easily communicate and integrate. Promote experimentation by encouraging the team to experiment with different integration approaches and learn from each iteration. Emphasize the value of incremental progress over perfection.

### **Talent & Skill Gaps**

Implement continuous learning programs with a focus on both technical skills (AI, machine learning, automation) and soft skills (collaboration, communication). Utilize online courses, workshops, and certifications. Encourage cross-functional training where testers learn AI concepts and AI specialists understand testing processes. Establish a mentorship program where experienced professionals guide those less familiar with AI and automation. Promote peer learning sessions to share knowledge. Cultivate a culture where learning new skills is valued and celebrated. Emphasize that skill development is a journey, and setbacks are part of the learning process.

### **Cultural and Organizational Resistance**

Implement a structured change management strategy to communicate the benefits of shift-left and AI adoption. Involve key stakeholders early and address concerns transparently. Use collaborative tools to break down silos and foster communication across teams. Regular cross-department meetings can help align goals and improve understanding. Start with pilot projects to demonstrate the value of shift-left and AI, gradually expanding as teams gain confidence and see the benefits firsthand. Create an environment where team members feel safe to express concerns and ask questions. Use feedback to iteratively improve processes and address resistance constructively.

### **Cost & Resource Allocation**

Focus on AI and automation initiatives with a clear return on investment (ROI). Prioritize projects that provide quick wins and demonstrate value early on. Implement AI and automation in phases, starting with the most critical areas to manage costs and resources more effectively. Leverage open-source AI and automation tools to reduce costs. Invest in proprietary solutions only when necessary. Encourage the team to view AI and automation as long-term investments. Emphasize the importance of resilience and adaptability in overcoming short-term financial and resource challenges.

## Security & Compliance

Implement AI-driven security tools that can continuously monitor data usage, detect anomalies, and enforce compliance with regulations. Use automation tools to ensure continuous compliance with industry standards and regulations and reduce the burden on human resources. Conduct regular security audits using both AI and human expertise to identify potential vulnerabilities and rectify them proactively. Encourage the team to take ownership of security practices and view compliance as an integral part of their responsibilities. Foster a culture where security is a shared priority.

By applying these strategies, you can effectively address the challenges associated with AI and automation in software development, ensuring that the team is well-equipped to handle the complexities of the modern SDLC while maintaining a growth mindset.

## 6 Conclusion

Resolving the defects in production can cost multiple times more than addressing them earlier in the software development lifecycle (SDLC). The Shift Left Testing approach aims to identify defects as early as possible in the SDLC, resulting in reduced costs, effort, and time spent on rectifying defects, increased efficiency, improved product quality, and shorter time-to-market.

To achieve these benefits, we are developing a transformative testing solution that seamlessly facilitates adopting the Shift Left Testing approach. This solution leverages the integration of Artificial Intelligence (AI) and automation capabilities to foster a proactive, efficient, and high-quality software development process.

## References

Shreya Bose. Feb 9, 2023. "Shift Left Testing: What It Means and Why It Matters", <https://www.browserstack.com/guide/what-is-shift-left-testing>

Testim. June 11, 2021. "What is Shift Left Testing? A Guide to Improving Your QA", <https://www.testim.io/blog/shift-left-testing-guide/#:~:text=The%20%E2%80%9Cshift%20left%E2%80%9D%20testing%20movement,phase%20that%20require%20code%20patching.>

Anushtha Jain. July 6, 2023. "How Can AI Enhance Requirements Management Processes and Outcomes?", <https://www.linkedin.com/pulse/how-can-ai-enhance-requirements-management-processes/>

Hannah Clark. "The Software Development Life Cycle (SDLC): 7 Phases and 5 Models", <https://theproductmanager.com/topics/software-development-life-cycle/>

Matt Young. Sept 24, 2018. "How AI Enables Shift Left Testing For Software Teams", <https://www.functionize.com/blog/ai-enables-teams-shift-left-software-testing#:~:text=There%20are%20other%20intriguing%20ways,human%20would%20be%20able%20to.>

ZYMR. "4 Major Benefits of AI For Shift Left Testing", <https://www.zymr.com/infographic/4-major-benefits-of-ai-for-shift-left-testing>

Atimi Software, June 7, 2023. "The Advantages and Challenges of Shift-Left Testing (and how to incorporate it into your development process)", <https://www.linkedin.com/pulse/advantages-challenges-shift-left-testing-how-incorporate/>

# Secure Software, Inside and Out:

## Seven Steps

**Joye Purser, CISSP, PhD; Jan Graves, CISSP; Walter Angerer**

[Joye.Purser@Veritas.com](mailto:Joye.Purser@Veritas.com)

### Abstract

As a 40-year software industry veteran and 18-consecutive-year Gartner Magic Quadrant winner for data protection, Veritas Technologies constantly works to generate secure software. Leveraging CISA's "Secure by Design" principles for developers, Veritas incorporates people, processes, and technology to generate software products that are secure.



Incidents such as the SolarWinds attack presented an inflection point for Silicon Valley. The supply chain, or sequence of tools and activities employed to produce software products, has come under intense scrutiny. In this paper, we explain seven key phases of our SecDevOps process and how we employ chaos engineering to generate a robust product relied upon by 95% of global Fortune 100 companies.

In our discussion of 7 key steps in a mature SecDevOps process, we explain why chaos engineering techniques yield a better software product. Our Veritas REDLab is a unique, internal 'cyber range' where wild type malware is released into an isolated, test environment. This and other 'stress test' techniques harden our software during multiple points of the build process. True to our California roots, we build fun into the process and thus contribute to a positive company culture that retains talent and bolsters security.

### Biography

*Dr. Joye Purser is the Global Field CISO at Veritas Technologies. She consults with security leaders globally on all manner of cybersecurity topics. Before that, Dr. Purser served 18 years in the federal government in senior security executive roles, including the White House and Pentagon. She is the ISC2 Tipton Lifetime Achievement Award Winner for Information Security.*

*Jan Graves is the Chief Product Security Officer at Veritas. In this role he leads the organization responsible for the Application Security Assurance Program, product certification roadmaps, and oversees all aspects of secure software development. Prior to Veritas, Mr. Graves was Lead Systems Engineer for over a decade at Innovative Computer Systems, Inc.*

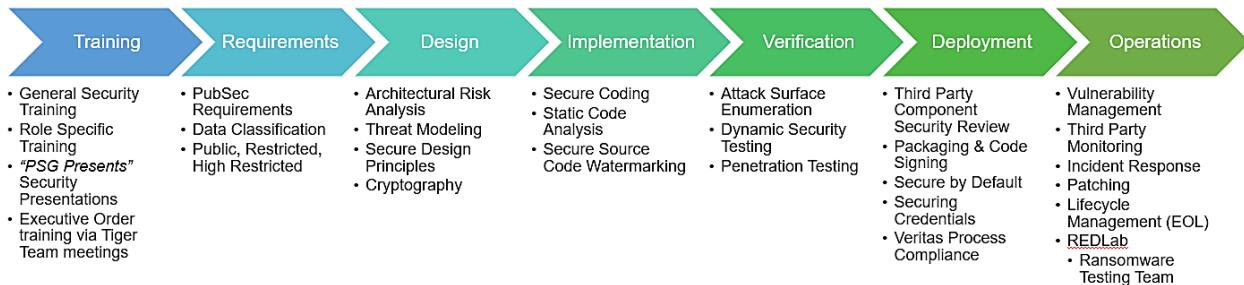
*Walter Angerer is Senior Vice President of Engineering at Veritas and frequent public speaker on secure software development techniques. Before that, Mr. Angerer was CEO and Board Member of Parsec Labs, LLC, Quorum, and other data storage innovative startup companies. Earlier career experiences include software development and management at Siemens Power Transmission and Distribution.*

# 1 Introduction

Software code developers have become modern day armor-makers. Cyber adversaries aggressively target code vulnerabilities. Both militaries and champion video gamers know that (a) knowledge of adversary vulnerabilities offers tactical advantage; and (b) defenders must know and understand their own vulnerabilities and aggressively control and protect such details among those with “need to know.”

Today’s software developers, teams, and leaders must concern themselves with much more than Agile processes. With cyber threat actors searching for any opportunity to exploit code vulnerabilities, the code itself must be developed securely. Robust information protection measures must be in place.

Thus, the concept of SecDevOps was born. This simply means that the processes by which software is coded and developed must include security at all stages. People, processes, and technology must align and work in close coordination to deliver “defense in depth” during the software development lifecycle: from design, code development, assembly, package, test, deploy and operation-and-maintenance (O&M). Teams are using chaos engineering techniques to test that code so it that is ‘hardened’ against the threat actors of today and tomorrow.



# 2 A Program for Application Security

The Veritas Application Security Assurance Program (ASAP) is based on Microsoft’s Secure Development Lifecycle for Agile Developers<sup>1</sup>. The program consists of seven, non-linear pillars. The pillars represent people, processes, and tools focused on a major phase or component of the software development lifecycle.

Over time, Veritas has evolved its methodology to address real-world challenges and requirements of *U.S. Presidential Executive Order 14028*<sup>2</sup>. When the Order was announced, we verified that we already followed the majority of the Secure Software Development Framework (SSDF) as outlined in NIST SP 800-218<sup>3</sup>. The Veritas Product Security Group (PSG) incorporates security practices throughout the entire development lifecycle for each software product. Veritas uses a variety of automated security tools as well as manual techniques to find vulnerabilities in our products in accordance with Executive Order and CISA requirements. Durable, dedicated teams meet with PSG bi-weekly on Application Security; Monthly on PubSec and Executive Order 14028 compliance.

## 2.1 Training

Veritas ensures that product development teams perform regular security awareness training to stay up to date with evolving security threats and mitigation techniques. Training also incorporates role-based scenarios to increase awareness of attack vectors and secure coding techniques. We incorporate chaos engineering techniques in our cyber-range awareness training: developers play the attacker’s role to better understand threats and mitigations under a variety of ever-changing scenarios.

## 2.2 Requirements and Compliance

A second pillar of our Application Security Assurance Program includes a program to track and monitor security and performance requirements. These requirements originate from our own internal standards, as well as regulations emanating from public sector and highly regulated customers such as financial organizations. Like a Governance, Risk, and Compliance (GRC) organization, our teams address the business case and cost implications of a large number of regulatory requirements. Our specialists evaluate that products follow secure design principles, such as least privilege, defense in depth, and secure defaults. They assess that software products have undergone threat modeling to identify potential security risks and vulnerabilities. They act as independent, internal auditors to monitor enforcement of Executive Order 14028 for items such as multi-factor authentication, encryption for data at rest and in transit (FIPS 140-2)<sup>4</sup>, Zero Trust architecture, and software bill of materials (SBOMs). The teams also align best practices and security requirements with the SSDF.

## 2.3 Secure Software Design

Veritas works to ensure that secure design principles are met. Our Product Security Group assesses if each product has undergone threat modeling to identify potential security risks and vulnerabilities. Product teams follow “Secure by Design” principles defined by the Cybersecurity and Infrastructure Security Agency (CISA)<sup>5</sup>. At the tactical level, such principles include the use of memory safe program languages. A secure hardware foundation is employed, considering the hardware supply chain down to the chip level. Static and dynamic testing; code review by peers; defense-in-depth principles; and CISA Cyber Performance Goals are incorporated into our framework.

Products adhere to relevant security standards, regulations, and best practices; these include ISO 27001<sup>6</sup>, the NIST Cybersecurity Framework, industry-specific security requirements

## 2.4 Implementation

The Implementation pillar includes processes to ensure product code reviews are conducted accurately and any issues identified have been addressed. Each product has undergone threat modeling to identify potential security risks and vulnerabilities. Products also have an extensive cryptography review as well as source code reviews, which seek to identify security flaws such as injection vulnerabilities, authentication bypass, or insecure data handling. We employ a control analysis tool is executed and that identified issues are addressed. Teams verify that the software product undergoes rigorous security testing. Teams address attack surface enumeration. We perform dynamic security testing and undergo penetration testing.

## 2.5 Deployment

This pillar validates that each product team enforces secure configuration management practices. We hire third parties to conduct independent Component Security Reviews. We conduct independent team oversight of packaging and code-signing processes. This includes securing credentials and ensuring overall process compliance. Related to vendors and third-party assessment: the PSG works with each product team to ensure that all vendors assess their third-party components and dependencies for potential security vulnerabilities or weaknesses. We enforce secure configuration management practices, such as storing credentials securely, protecting sensitive data, and implementing secure communication protocols.

## 2.6 Operations and Maintenance

In terms of product maintenance over time, we regularly assess and update our vulnerability policy for patching and other security-related code lifecycle management tasks. Veritas uses multiple automated security tools and manual techniques to find vulnerabilities in our products in accordance with Executive Order and CISA requirements. Veritas scores the severity of vulnerabilities using an industry standard system (Common Vulnerability Scoring System v3.1) to guide remediation urgency and public notifications. Dedicated product security leads meet with the Product Security Group bi-weekly to review application security topics. Veritas releases monthly security patches to ensure that customers have the latest fixes.

Product releases are not made available unless the PSG has approved a software bill of materials (SBOM). Veritas PSG generated a tool that uses heuristics to do basic validation of the SBOM delivered. The tool provides views of the data to simplify human review. The tool also identifies issues for reporting in the Plan of Action and Milestones. The product team and PSG review the tool results; human intervention is crucial in generating an SBOM. The SBOMs are delivered in machine-readable format, specifically, Software Package Data Exchange (SPDX)<sup>7</sup>. Veritas is exploring third-party validations for SBOMs and attestations for product compliance with the SSDF.

The Operations element validates that each product team has an established incident response plan, including procedures for handling security incidents, incident reporting, and communication channels.

## 3 REDLab

Veritas has also built REDLab<sup>8</sup>, an isolated lab to study ransomware and malware attacks firsthand. We built Veritas REDLab, a proprietary, secure facility purpose-built to study the impact of live malware on our products. REDLab provides an isolated environment air-gapped from all Veritas network infrastructure. The lab uses live malware and ransomware variants to simulate real-world scenarios, testing our solutions to ensure it is hardened against modern threats at the technical level. We conduct our own research to study attacks as they occurred so we can (a) assess features to aid in detecting ransomware attacks; (b) improve protection of backup repositories; and (c) provide faster recovery when needed.

Staffed by senior security engineers from several established security organizations, REDLab is isolated from the outside world. Chaos engineering is employed to stress-test our software against a variety of adversary tactics, techniques, procedures (TTPs) and code variants. We have hired an external consulting team with more than 100 years of combined experience to validate our initial REDLab tests. These tests provided us with new insights into the inner workings of malware itself.

The REDLab initiative has improved our understanding of the requirements for infrastructure, applications, ransomware identification, and debugging. It also helped define how to simulate disaster recovery scenarios, as well as maintain, clean up, and quickly rebuild systems. The REDLab team synthesized learnings that contributed to ASAP improvements including: (a) implementing secure ways to bring in product binaries; (b) writing a product-specific fuzzer program to expose vulnerabilities; and (c) defining standard operating procedures to test and evaluation (T&E).

## 4 Conclusion

Signed the CISA Secure by Design Pledge is an important marker in a continuous evolution of our secure code development practice. Evolving our teams, skills, processes, and tools to meet tomorrow's software attacks is a high priority for us. Our ability to deliver data recovery in all-hazards scenarios is

paramount to our customers, and as critical IT infrastructure, we strive daily to produce software that is as clean, secure, and robust as possible.

It is our intent to share these practices with the broader developer community to build awareness and enhance our collective security.

## 5 Sources Cited

- 
- <sup>1</sup> Microsoft. "Microsoft Security Development Lifecycle (SDL)." Accessed September 11, 2024. <https://www.microsoft.com/en-us/securityengineering/sdl>.
  - <sup>2</sup> The White House. "Executive Order on Improving the Nation's Cybersecurity, May 12, 2021." Accessed September 11, 2024. <https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/>.
  - <sup>3</sup> NIST Computer Security Resource Center Special Publication 800-218. "Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities." Accessed September 11, 2024. <https://csrc.nist.gov/pubs/sp/800/218/final>.
  - <sup>4</sup> NIST Computer Security Resource Center FIPS 140-2. "Security Requirements for Cryptographic Modules." Accessed September 11, 2024. <https://csrc.nist.gov/pubs/fips/140-2/upd2/final>.
  - <sup>5</sup> CISA. "Secure by Design." Accessed September 11, 2024. <https://www.cisa.gov/securebydesign>
  - <sup>6</sup> ISO, the International Organization for Standardization. "ISO/IEC 27001:2022, Information Security, Cybersecurity, and Privacy Protection — Information Security Management Systems Requirements." Published Edition 3, 2022. Accessed September 11, 2024. <https://www.iso.org/standard/27001>.
  - <sup>7</sup> The Linux Foundation Projects. "System Package Data Exchange (SPDX®)." Accessed September 11, 2024. <https://spdx.dev/>.
  - <sup>8</sup> Veritas Technologies LLC. "Veritas Trust Center: REDLab." Accessed September 11, 2024. <https://www.veritas.com/why-veritas/trust/redlab>.

# Quality Maturity Model: Driving Excellence in High-Growth Orgs

Vidhya Ranganathan

[vidhya.ranganathan@okta.com](mailto:vidhya.ranganathan@okta.com)

## • Abstract

In high-growth organizations, ensuring product quality amidst rapid development cycles is critical. While frameworks like DORA (DevOps Research and Assessment) and SPACE (Software Productivity, Assessment, Control, and Evaluation) have advanced developer productivity, a significant gap exists in systematically improving quality practices. This paper presents a comprehensive Quality Maturity Model specifically for high-growth organizations. The model encompasses fourteen quality dimensions, ranging from static testing to configuration management, providing a holistic approach to quality enhancement. Through case studies and implementation strategies, we demonstrate the model's efficacy in improving product quality, reducing defects, and increasing operational efficiency in rapidly scaling environments.

## • Biography

Vidhya Ranganathan, with 14 years of experience in the quality domain, began her career as a manual tester and developed a passion for automation and DevOps. Specializing in designing frameworks, curating processes, and forming dedicated quality teams, she has significantly contributed to several startups, helping them bootstrap and scale their quality teams. Currently a Quality Manager at Okta, Vidhya is known for her strategic approach and hands-on expertise in driving excellence in quality assurance.

# 1 Introduction

Maintaining product quality while keeping up with rapid development cycles presents a formidable challenge in the fast-paced world of high-growth organizations. As companies scale at unprecedented rates, traditional quality assurance (QA) practices often struggle to keep pace with the evolving demands of the market and the complexity of modern software systems.

High-growth organizations are characterized by their rapid scaling, frequent product iterations, and the need for agility in responding to market demands. These characteristics create a unique set of challenges for quality management. However, it's crucial to recognize that the approach to software quality in these organizations is not monolithic; it varies significantly based on the current priorities and stage of growth of each organization.

Several factors influence how high-growth organizations approach software quality:

**1. Growth Stage:**

- Early-stage startups often prioritize speed to market and feature development over comprehensive quality processes.
- Mid-stage companies may begin to feel the pain of technical debt and start investing more in quality infrastructure.
- Later-stage organizations typically have more resources to dedicate to robust quality practices but may struggle with legacy systems.

**2. Market Pressure:**

- In highly competitive markets, companies may prioritize rapid feature delivery to stay ahead, potentially compromising on thorough quality checks.
- In regulated industries (e.g., fintech, healthcare), compliance requirements may necessitate stricter quality controls regardless of growth stage.

**3. Product Type:**

- B2C products might focus more on user experience and performance, prioritizing these aspects of quality.
- B2B or enterprise software may emphasize reliability, security, and scalability in their quality efforts.

**4. Funding Situation:**

- Well-funded startups might have the luxury to invest in quality infrastructure early on.
- Bootstrapped companies might need to be more selective in their quality initiatives due to resource constraints.

**5. Technical Leadership:**

- The background and philosophy of technical leaders significantly influence the emphasis placed on different aspects of quality.
- Some leaders might prioritize automated testing and CI/CD, while others might focus on code reviews and architectural quality.

**6. Customer Base:**

- B2B companies with a few large enterprise customers might prioritize stability and security in their quality efforts.
- B2C companies with a large user base might focus more on performance and user experience.

#### 7. Scalability Needs:

- Organizations experiencing or anticipating hypergrowth might prioritize scalability and performance in their quality initiatives.
- Companies with more predictable growth might have the luxury to focus on a broader range of quality aspects.

#### 8. Recent Incidents:

- A major security breach might shift focus towards security testing and hardening.
- Performance issues during peak times might prioritize load testing and performance optimization.

These varying priorities create a complex landscape for quality management in high-growth organizations.

## 2 Objectives

The primary objective of this paper is to introduce a Quality Maturity Model (QMM) specifically designed for high-growth organizations. This model aims to provide a comprehensive, flexible, and actionable framework for quality enhancement. The key objectives are:

### **Comprehensive and Customizable Quality Framework**

- Provide a multidimensional framework covering various aspects of software development and operations
- Define clear maturity levels from "Bronze" to "Platinum" for each dimension
- Enable customization to fit specific organizational needs, industries, and growth stages
- Offer a "North Star" vision of excellence for each quality dimension
- Bridge gaps in existing frameworks like DORA and SPACE with a quality-focused perspective

### **Data-Driven Quality Enhancement and Scalability**

- Facilitate data-driven decision-making with key metrics for each maturity level
- Balance speed and quality, showing how advanced practices can accelerate development
- Support scalability as organizations grow from startup to enterprise-level
- Enable benchmarking against industry standards and tracking progress over time
- Anticipate future needs and prepare for emerging technologies and methodologies

### **Cultural Transformation and Organizational Alignment**

- Promote a culture where quality is everyone's responsibility
- Provide a common language and shared goals across teams and departments
- Encourage a shift towards proactive quality management
- Support the creation of customized quality roadmaps aligning with organizational priorities
- Foster continuous improvement and innovation in quality practices

By achieving these objectives, the Quality Maturity Model aims to provide high-growth organizations with a powerful tool for systematically enhancing their quality practices. The model's emphasis on defined maturity levels with specific qualifiers allows organizations to create a customized quality roadmap that aligns with their unique circumstances and aspirations. Moreover, by offering a clear maturity level for each quality dimension, the model motivates teams to continuously improve, fostering a culture of quality excellence. This approach not only helps organizations address their immediate quality challenges but also prepares them for sustainable, long-term growth while maintaining high standards of product quality.

### 3 Literature Review

The landscape of software development and operations has seen significant advancements in recent years, with frameworks and methodologies aimed at improving various aspects of the development lifecycle. However, most of these frameworks focus primarily on delivery optimization and developer productivity, leaving a gap in comprehensive quality management, especially for high-growth organizations.

#### Existing Frameworks

##### DORA (DevOps Research and Assessment)

The DORA framework, introduced by [Forsgren et al. \(2018\) in their book "Accelerate: The Science of Lean Software and DevOps,"](#) focuses on four key metrics:

- Deployment Frequency: How often an organization successfully releases to production
- Lead Time for Changes: The time it takes to go from code committed to code successfully running in production
- Change Failure Rate: The percentage of deployments causing a failure in production
- Time to Restore Service: How long it takes to restore service when a failure occurs in production

##### SPACE Framework

The SPACE framework, introduced by [Forsgren et al. \(2021\) in their paper "The SPACE of Developer Productivity,"](#) offers a multidimensional view of developer productivity:

- Satisfaction and well-being
- Performance
- Activity
- Communication and Collaboration
- Efficiency and flow

#### Gaps in Existing Literature

Several key gaps emerge when examining the existing literature and frameworks:

- Focus on Delivery over Quality
- Insufficient Attention to Scaling Challenges
- Limited Integration of Quality Across the Lifecycle

## 4 Quality Maturity Model

The proposed Quality Maturity Model consists of fourteen dimensions, each critical for maintaining high standards of product quality. These dimensions cover the entire software development lifecycle, from code creation to production management, providing a comprehensive framework for quality assessment and improvement. A key feature of this model is its flexibility and adaptability to different organizational contexts. While the model provides a structured approach to quality improvement, it intentionally avoids prescribing rigid, one-size-fits-all definitions for each maturity level. This design choice recognizes that high-growth organizations operate in diverse environments, with varying priorities, resources, and constraints.

The model uses a four-level maturity scale for each dimension:

1. Bronze: Basic practices are in place, but they may be inconsistent or reactive.
2. Silver: Consistent practices are established and proactively applied.
3. Gold: Advanced practices are implemented, with a focus on optimization and integration.
4. Platinum: Industry-leading practices are in place, with continuous innovation and improvement.

However, what constitutes "Bronze" or "Platinum" for a particular dimension can vary between organizations. For instance, a small startup might consider basic automated unit testing as a "Silver" achievement, while a large enterprise might view this as a "Bronze" baseline. The model encourages organizations to interpret and adapt these levels based on their specific context, industry standards, and growth stage. This flexibility allows organizations to:

- **Customization to Align with Goals:** Organizations can tailor the QMM to reflect their specific industry requirements and strategic priorities. This allows them to focus on quality dimensions most relevant to their products or services. By doing so, they ensure that quality improvements directly contribute to their overall business objectives and competitive advantage.
- **Prioritization of Critical Dimensions:** Companies can identify and focus on the quality dimensions that have the most immediate impact on their business. This targeted approach allows for efficient resource allocation and quicker realization of benefits. It enables organizations to address their most pressing quality challenges first, creating a foundation for broader improvements.
- **Realistic Target Setting:** The model's maturity levels provide a clear progression path, allowing organizations to set achievable short-term and long-term goals. This step-by-step approach prevents overwhelm and maintains team motivation. It also allows for celebratory milestones, reinforcing the importance of quality improvement efforts.
- **Benchmarking Against Peers:** Organizations can compare their quality practices with similar companies, providing valuable context for their maturity levels. This benchmarking helps identify areas where they're leading or lagging in their industry. It also offers insights into best practices and potential areas for improvement based on peer performance.
- **Adaptation as Organization Grows:** The flexibility of the QMM allows it to evolve alongside the organization, accommodating changes in size, complexity, and market position. As the company grows, it can adjust its quality targets and introduce more sophisticated practices. This ensures that quality management remains relevant and effective throughout the organization's lifecycle.

## Dimension Descriptions

Dimension	Business Considerations	Recommended metrics
<b>Static Testing</b>  Static testing involves analyzing code without execution to identify potential issues early in the development process.	- Early detection of code issues reduces downstream costs - Improves code consistency and maintainability - Helps in enforcing coding standards across the team	<u>Bronze</u> : Basic linting rules applied, <50% code coverage  <u>Silver</u> : Automated analysis in CI/CD, 50-75% coverage  <u>Gold</u> : Advanced tools (security, performance), 75-90% coverage  <u>Platinum</u> : 90%+ coverage, AI-assisted analysis, custom rules
<b>Functional Testing</b>  Manual testing involves human testers executing test cases and exploring the application to find defects and usability issues.	- Essential for user experience validation - Helps identify usability issues that automated tests might miss - Important for complex scenarios and exploratory testing	<u>Bronze</u> : Ad-hoc testing before releases  <u>Silver</u> : Structured test plans, regular testing cycles  <u>Gold</u> : Comprehensive plans, risk-based approach  <u>Platinum</u> : Continuous exploratory testing, usability studies
<b>Unit and Integration Testing</b>  Unit testing involves testing individual components, while integration testing checks the interaction between components. These testing types ensure that both individual parts and their interactions work as expected, catching issues early in the development process.	- Foundational for code reliability and maintainability - Enables confident refactoring and feature additions - Supports faster development cycles	<u>Bronze</u> : Basic unit tests for critical components  <u>Silver</u> : 60-70% code coverage, CI integration  <u>Gold</u> : 80-90% coverage, advanced mocking  <u>Platinum</u> : 90%+ coverage, property-based testing  <b><i>Unit and integration testing are combined in this model as they form the foundation of the testing pyramid, often implemented at similar stages in the development process. This combination encourages a holistic approach to low-level testing, ensuring both</i></b>

		<b><i>granular and interconnected functionalities are verified efficiently.</i></b>
<b>Use Case Documentation</b>  Use case documentation involves detailing user interactions and system behaviors for various scenarios. Well-documented use cases ensure that development and testing efforts align with user needs and business requirements.	- Aligns development with business requirements - Improves communication between stakeholders - Serves as a basis for test planning and user acceptance testing	<u>Bronze</u> : Basic documentation for key features  <u>Silver</u> : Structured use cases for all major features  <u>Gold</u> : Comprehensive use cases with regular reviews  <u>Platinum</u> : Living documentation, auto-updated from code
<b>System Testing</b>  System testing evaluates the complete and integrated software system against specified requirements.	- Ensures the entire system works as intended - Crucial for identifying integration issues - Validates end-to-end functionality	<u>Bronze</u> : Manual system testing before major releases <u>Silver</u> : Automated tests for critical paths <u>Gold</u> : Comprehensive automated suite, regular full regression <u>Platinum</u> : 85% of use cases automated
<b>Performance Testing</b>  Performance testing assesses the system's responsiveness, stability, and scalability under various load conditions.  Performance testing ensures that the application can handle expected loads and degrade gracefully in cases of load spikes, providing a good user experience and avoiding system failures.	- Ensures the system can handle expected and peak loads - Critical for user satisfaction and system reliability - Important for capacity planning and scaling decisions	<u>Bronze</u> : Basic load testing for critical paths  <u>Silver</u> : Regular performance tests with defined SLAs  <u>Gold</u> : Comprehensive performance suite, stress testing  <u>Platinum</u> : Continuous performance monitoring, predictive analysis
<b>Release Safety</b>	- Minimizes the risk of deployment-	<u>Bronze</u> : Manual deployment

<p><b>Release safety</b> focuses on minimizing risks associated with deploying new code to production.</p>	<p>related issues</p> <ul style="list-style-type: none"> <li>- Ensures business continuity during updates</li> <li>- Supports faster and more frequent releases</li> </ul>	<p>checklists</p> <p><u>Silver</u>: Automated deployment pipelines</p> <p><u>Gold</u>: Blue-green deployments, automated rollback</p> <p><u>Platinum</u>: Canary releases, feature flags, automated verification</p>
<p><b>Observability</b> Observability allows teams to understand and debug system behavior in production through logging, monitoring, and tracing.</p>	<ul style="list-style-type: none"> <li>- Enables quick identification and resolution of issues</li> <li>- Provides insights for system optimization</li> <li>- Supports proactive problem prevention</li> </ul>	<p><u>Bronze</u>: Basic error logging and monitoring</p> <p><u>Silver</u>: Centralized logging, basic alerting</p> <p><u>Gold</u>: Distributed tracing, advanced alerting</p> <p><u>Platinum</u>: Anomaly detection, predictive analytics</p>
<p><b>Operational Resilience</b> Operational resilience focuses on the system's ability to handle and recover from failures. Resilient systems can maintain functionality in the face of errors or increased load, ensuring better uptime and user satisfaction.</p>	<ul style="list-style-type: none"> <li>- Ensures the system can handle and recover from failures</li> <li>- Critical for maintaining service levels and user trust</li> <li>- Important for businesses with high availability requirements</li> </ul>	<p><u>Bronze</u>: Basic failover processes</p> <p><u>Silver</u>: Automated failover, regular resilience testing</p> <p><u>Gold</u>: Self-healing systems, chaos engineering practices</p> <p><u>Platinum</u>: Predictive resilience, automated mitigation strategies</p>
<p><b>Incident Management</b> Incident management involves the processes for detecting, responding to, and learning from production issues.</p>	<ul style="list-style-type: none"> <li>- Minimizes impact of production issues on users and business</li> <li>- Improves the team's ability to handle and learn from incidents</li> <li>- Critical for maintaining service quality and reliability</li> </ul>	<p><u>Bronze</u>: Basic incident response process</p> <p><u>Silver</u>: Defined roles and escalation procedures</p> <p><u>Gold</u>: Post-mortems, incident learnings incorporated into processes</p> <p><u>Platinum</u>: Data-driven Incident prediction and prevention</p>
<p><b>Disaster Recovery</b> Disaster recovery planning ensures business continuity in the event of major system failures or catastrophes.</p>	<ul style="list-style-type: none"> <li>- Ensures business continuity in catastrophic scenarios</li> <li>- Critical for data protection and regulatory compliance</li> <li>- Important for maintaining stakeholder trust</li> </ul>	<p><u>Bronze</u>: Basic backup system</p> <p><u>Silver</u>: Regular DR drills, documented recovery procedures</p> <p><u>Gold</u>: Automated failover to DR site, RPO/RTO optimization</p> <p><u>Platinum</u>: Multi-region active-active setup, continuous DR testing</p>
<p><b>Security Testing</b></p>	<ul style="list-style-type: none"> <li>- Protects against data breaches</li> </ul>	<p><u>Bronze</u>: Annual penetration testing</p>

<p>Security testing involves assessing the system for vulnerabilities and ensuring that security measures are effective.</p>	<p>and cyber attacks</p> <ul style="list-style-type: none"> <li>- Critical for maintaining user trust and regulatory compliance</li> <li>- Important for protecting intellectual property</li> </ul>	<p><u>Silver</u>: Regular vulnerability scans, secure coding practices  <u>Gold</u>: Continuous security testing, threat modeling  <u>Platinum</u>: Bug bounty program, AI-driven security analysis</p>
<p><b>Secret Management</b> Secret management involves securely storing, accessing, and rotating sensitive information such as API keys and passwords.</p>	<ul style="list-style-type: none"> <li>- Prevents unauthorized access to sensitive data and systems</li> <li>- Critical for maintaining security and compliance</li> <li>- Important for managing access across growing teams</li> </ul>	<p><u>Bronze</u>: Encrypted storage of secrets  <u>Silver</u>: Centralized secret management system  <u>Gold</u>: Automated secret rotation, access auditing  <u>Platinum</u>: Zero-trust architecture, access anomaly detection</p>
<p><b>Configuration Management</b> Configuration management involves maintaining consistent system configurations across different environments.</p>	<ul style="list-style-type: none"> <li>- Ensures consistency across environments</li> <li>- Reduces "works on my machine" issues</li> <li>- Supports faster and more reliable deployments</li> </ul>	<p><u>Bronze</u>: Manual configuration tracking  <u>Silver</u>: Version-controlled configurations  <u>Gold</u>: Infrastructure as Code for all environments  <u>Platinum</u>: Automated compliance checks, self-service provisioning</p>

## 5 Implementation and Case Studies

To demonstrate the practical application and effectiveness of the Quality Maturity Model (QMM), we present two case studies of companies that implemented the model with different approaches and priorities.

### Key Implementation Details:

	Early Stage Startup	High Growth Startup
<p><b>Business Considerations</b></p>	<ul style="list-style-type: none"> <li>• Critical nature of the product: Cybersecurity products require extremely high reliability and security.</li> <li>• Enterprise customers: Fortune 500 clients have high expectations and stringent requirements.</li> <li>• Reputation-driven market: Success heavily depends on building and maintaining trust.</li> <li>• Limited resources: With only 50</li> </ul>	<ul style="list-style-type: none"> <li>• Rapid scaling: Need for quality practices that can scale with 200% year-over-year growth</li> <li>• Expanding team: Quality practices must be easily adoptable by new team members</li> <li>• Product reliability: Maintaining product quality during rapid growth is crucial for customer retention</li> </ul>

	<p>employees, efficiency in quality processes is crucial.</p> <ul style="list-style-type: none"> <li>Potential for high-impact failures: A single major bug could severely damage the company's reputation.</li> </ul>	<ul style="list-style-type: none"> <li>Diverse customer base: Need to cater to varying quality expectations across a broad customer spectrum</li> <li>Speed of innovation: Quality practices should not significantly slow down the pace of feature development</li> </ul>
<b>Investment Philosophy</b>	<ul style="list-style-type: none"> <li>Slow but steady progress and improvements</li> <li>Focus on building a strong foundation of quality practices</li> <li>Emphasis on manual processes and expert oversight</li> <li>Willingness to invest time in thorough, hands-on quality assurance</li> <li>Prioritize critical areas first, gradually expanding comprehensive quality practices</li> </ul>	<ul style="list-style-type: none"> <li>Data-driven, low-touch approach</li> <li>Focus on automation and scalable processes</li> <li>Willingness to invest in tools and infrastructure that support rapid scaling</li> <li>Emphasis on metrics and quantifiable improvements</li> <li>Balance between maintaining quality and supporting rapid growth</li> </ul>
<b>Approach to QMM</b>	<ul style="list-style-type: none"> <li>Start with basic practices across all dimensions</li> <li>Invest heavily in security testing and secret management given the nature of their product</li> <li>Implement thorough manual testing processes</li> <li>Gradually introduce automation in critical areas</li> <li>Focus on building robust incident management and disaster recovery processes</li> </ul>	<ul style="list-style-type: none"> <li>Implement automated testing across all levels (unit, integration, system)</li> <li>Invest in robust observability and monitoring tools</li> <li>Develop data-driven performance testing practices</li> <li>Implement automated deployment pipelines with strong release safety measures</li> <li>Focus on configuration management and secret management to support rapid scaling</li> <li>Develop self-service tools for quality processes to reduce bottlenecks</li> </ul>
<b>Goals</b>	<p><b>Short Term (6 months):</b></p> <ul style="list-style-type: none"> <li>Achieve Bronze level in at least 10 out of 14 QMM dimensions. This was important as they were getting started with the quality practice.</li> <li>Reach Silver level in Security Testing and Secret Management</li> <li>Reduce critical bugs reaching production by 20%</li> </ul>	<p><b>Short Term (6 months):</b></p> <ul style="list-style-type: none"> <li>Achieve Silver level in at least 10 out of 14 QMM dimensions</li> <li>Reach Gold level in Release Safety and Configuration Management</li> <li>Reduce deployment-related incidents by 30%</li> <li>Increase development</li> </ul>

	<ul style="list-style-type: none"> <li>• Improve customer satisfaction scores by 10%</li> </ul> <p><b>Long-term Goals (12-14 months):</b></p> <ul style="list-style-type: none"> <li>• Achieve Silver level in at least 10 out of 14 QMM dimensions</li> <li>• Reach Gold level in Security Testing and Secret Management</li> <li>• Reduce critical bugs reaching production by 35%</li> <li>• Improve customer satisfaction scores by 20%</li> </ul>	<ul style="list-style-type: none"> <li>• velocity by 15%</li> <li>• Maintain current growth rate while improving product stability metrics by 20%</li> </ul> <p><b>Long-term Goals (12-14 months):</b></p> <ul style="list-style-type: none"> <li>• Achieve Gold level in at least 10 out of 14 QMM dimensions</li> <li>• Reach Platinum level in Release Safety and Configuration Management</li> <li>• Reduce deployment-related incidents by 50%</li> <li>• Increase development velocity by 25%</li> <li>• Maintain aggressive growth targets while improving product stability metrics by 40%</li> </ul>
<b>Static Testing</b>	Bronze - Manual code reviews; Platinum - Automated static analysis integrated into CI/CD pipeline.	Bronze - 50% code coverage by static analysis tools; Platinum - 100% code coverage with custom rule sets.
<b>Functional Testing</b>	Bronze - Ad-hoc testing before releases; Platinum - Structured exploratory testing sessions with documented results.	Bronze - 70% of features manually tested before release; Platinum - 100% feature coverage with a risk-based testing approach.
<b>Unit and Integration Testing</b>	Bronze - Basic unit tests for critical components; Platinum - 90% coverage.	Bronze - 60% code coverage; Platinum - 90% code coverage for unit and integration tests.
<b>Use Case Documentation</b>	Bronze - Informal user stories; Platinum - Detailed use cases with regular stakeholder reviews.	Bronze - 50% of features with documented use cases; Platinum - 100% feature coverage with regularly updated use cases.
<b>System Testing</b>	Bronze - Manual system testing before major releases; Platinum - Automated end-to-end testing suite.	Bronze - 70% of critical paths covered by automated tests; Platinum - 100% critical path coverage with weekly full regression runs.
<b>Performance Testing</b>	Bronze - Basic load testing; Platinum - Regular performance testing with defined SLAs.	Bronze - Load Tests at a cadence relevant to deployments; Platinum - Daily performance tests with automated analysis and alerting.
<b>Release Safety</b>	Bronze - Manual checklists for	Bronze - 50% of deployments using

	deployments; Platinum - Automated deployment pipelines with rollback capabilities.	CI/CD pipeline; Platinum - 100% automated deployments with canary releases.
<b>Observability</b>	Bronze - Basic error logging; Platinum - Comprehensive monitoring and alerting system.	Bronze - Basic metrics and logging; Platinum - Full-stack observability with anomaly detection
<b>Operational Resilience</b>	Bronze - Manual failover processes; Platinum - Automated failover and self-healing systems.	Bronze - 99% uptime; Platinum - 99.99% uptime with automated failover and recovery.
<b>Incident Management</b>	Bronze - Reactive incident response; Platinum - Proactive incident prevention and post-mortem culture.	Bronze - 4-hour average time to resolve critical incidents; Platinum - 30-minute average resolution time with automated root cause analysis.
<b>Disaster Recovery</b>	Bronze - Basic backup system; Platinum - Fully tested and automated disaster recovery process.	Bronze - Monthly DR drills; Platinum - Weekly chaos engineering exercises with automated recovery.
<b>Security Testing</b>	Bronze - Annual penetration testing; Platinum - Continuous security scanning and regular third-party audits.	Bronze - Quarterly vulnerability scans; Platinum - Daily automated security scans with quick fix.
<b>Secret Management</b>	Bronze - Encrypted storage of secrets; Platinum - Automated secret rotation and access controls.	Bronze - Centralized secret storage; Platinum - Fully automated secret rotation with zero-trust architecture.
<b>Configuration Management</b>	Bronze - Manual configuration tracking; Platinum - Infrastructure as Code with version control.	Bronze - 80% of configs in version control; Platinum - 100% Infrastructure as Code with automated compliance checks
<b>Scorecard</b>	<ul style="list-style-type: none"> <li>• Static Testing: Silver (70% - Automated static analysis, not fully integrated into CI/CD)</li> <li>• Manual Testing: Silver (75% - Structured testing sessions, partially documented)</li> <li>• Unit and Integration Testing: Silver (68% code coverage)</li> <li>• Use Case Documentation: Silver (70% - Detailed use cases, irregular stakeholder reviews)</li> <li>• System Testing: Silver (65% - Partially automated end-to-end testing)</li> <li>• Performance Testing: Bronze (55% - Regular load testing, no defined SLAs yet)</li> <li>• Release Safety: Silver (70% - Partially automated deployment</li> </ul>	<ul style="list-style-type: none"> <li>• Static Testing: Gold (90% code coverage with partially custom rule sets)</li> <li>• Manual Testing: Gold (90% feature coverage with risk-based approach)</li> <li>• Unit and Integration Testing: Gold (85% code coverage)</li> <li>• Use Case Documentation: Gold (90% feature coverage, regularly updated)</li> <li>• System Testing: Gold (95% critical path coverage, bi-weekly full regression)</li> <li>• Performance Testing: Gold (Thrice-weekly performance tests with automated analysis)</li> <li>• Release Safety: Platinum (100% automated</li> </ul>

	<ul style="list-style-type: none"> <li>pipelines)</li> <li>Observability: Bronze (55% - Basic monitoring, limited alerting)</li> <li>Operational Resilience: Silver (60% - Some automated failover, no self-healing yet)</li> <li>Incident Management: Silver (70% - Reactive with some proactive measures)</li> <li>Disaster Recovery: Silver (65% - Tested recovery process, not fully automated)</li> <li>Security Testing: Gold (85% - Regular security scanning, occasional third-party audits)</li> <li>Secret Management: Gold (80% - Centralized management with partial rotation)</li> <li>Configuration Management: Silver (75% - Version control for most configs, not full IaC)</li> </ul>	<ul style="list-style-type: none"> <li>deployments with canary releases)</li> <li>Observability: Gold (85% full-stack observability, basic anomaly detection)</li> <li>Operational Resilience: Gold (99.95% uptime, automated failover)</li> <li>Incident Management: Gold (45-minute average resolution time, partial root cause automation)</li> <li>Disaster Recovery: Gold (Bi-weekly chaos engineering, partially automated recovery)</li> <li>Security Testing: Gold (Thrice-weekly automated scans, quick remediation)</li> <li>Secret Management: Gold (Automated rotation for most secrets, advancing towards zero-trust)</li> <li>Configuration Management: Platinum (100% Infrastructure as Code, automated compliance checks)</li> </ul>
<b>Results</b>	<p>After 8 months of implementing the QMM, they achieved a silver level across most dimensions and Gold in critical areas like Security Testing and Secret Management. They reported a 40% reduction in critical bugs reaching</p>	<p><b>Results:</b></p> <p>After implementing the QMM over a 12-month period, they achieved Gold level in most dimensions and Platinum in Release Safety and Configuration Management. They reported a 60% reduction in deployment-related incidents, a</p>

	production and a 25% increase in customer satisfaction scores.	30% increase in development velocity, and maintained their aggressive growth targets while significantly improving product stability.
--	--	---

#### **Key observations include:**

- Focusing on specific dimensions yields measurable improvements in related metrics
- The model's flexibility allows organizations to prioritize dimensions based on their unique needs. These organizations had a specific short-term and long-term vision of quality and where they wanted to take their quality practices. This helped in prioritization of the efforts
- Continuous assessment and improvement are crucial for sustaining quality enhancements as evidenced by both companies' experiences. Integrating constant assessment into the development process implicitly fosters a culture of continuous improvement, allowing teams to proactively identify and address quality issues. Moreover, maintaining a mature quality state requires substantial ongoing effort, especially in the face of constant code changes and evolving business needs. This underscores the importance of viewing quality as a dynamic, ever-evolving practice rather than a static achievement.

#### **Limitations and future work:**

- The model may require adaptation for different industries or organizational sizes
- Long-term studies are needed to assess the model's impact on sustained growth and quality
- Integration with existing frameworks like DORA and SPACE could provide a more comprehensive view of organizational performance

## **6 Conclusion**

The Quality Maturity Model presented in this paper offers a structured approach for high-growth organizations to enhance their quality practices systematically. By addressing multiple dimensions of quality, the model ensures comprehensive coverage and facilitates continuous improvement. The case studies demonstrate that focused improvements in specific dimensions can lead to significant enhancements in product quality, operational efficiency, and customer satisfaction. As organizations continue to scale rapidly, the ability to maintain and improve quality becomes increasingly critical.

Future research should focus on long-term studies of model implementation, industry-specific adaptations, and integration with other performance frameworks. By continuing to refine and expand this model, we can help high-growth organizations achieve and maintain excellence in their products and services while navigating the challenges of rapid scaling.

## ● References

- Burns, Brendan, Joe Beda, and Kelsey Hightower. *Kubernetes: Up and Running: Dive into the Future of Infrastructure*. Sebastopol, CA: O'Reilly Media, 2017.
- Cockburn, Alistair. *Writing Effective Use Cases*. Boston: Addison-Wesley, 2000.
- Craig, Rick D., and Stefan P. Jaskiel. *Systematic Software Testing*. Boston: Artech House, 2002.
- Forsgren, Nicole, Jez Humble, Gene Kim, and Mik Kersten. *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. Portland, OR: IT Revolution Press, 2018.
- Haeffner, Jennifer. *Site Reliability Engineering: How Google Runs Production Systems*. Sebastopol, CA: O'Reilly Media, 2017.
- Humble, Jez, and David Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Boston: Addison-Wesley, 2010.
- Jain, Raj. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. New York: Wiley, 1991.
- McGraw, Gary. *Software Security: Building Security In*. Boston: Addison-Wesley, 2006.
- Meszaros, Gerard. *xUnit Test Patterns: Refactoring Test Code*. Boston: Addison-Wesley, 2007.
- Poppendieck, Mary, and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Boston: Addison-Wesley, 2003.
- Red Hat, Inc. *Configuration Management with Ansible*. Raleigh, NC: Red Hat, Inc., 2017.
- Wood, Jamie. *Disaster Recovery, Crisis Response, and Business Continuity: A Management Desk Reference*. New York: Apress, 2014.

# Enhance Software Performance Testing with Artificial Intelligence

Rini Susan V S

rinisvs@gmail.com

## Abstract

With advances in data collection, processing, and computation, AI has become the latest buzzword in every industry. Performance testing is a type of testing in which the speed, responsiveness, and stability of a software, product, or network is evaluated under peak workload. Performance testing has evolved a lot over time. The focus has shifted from mere testing to performance engineering aspects in which the testing teams identify bottlenecks, perform error analysis, and provide performance tuning recommendations.

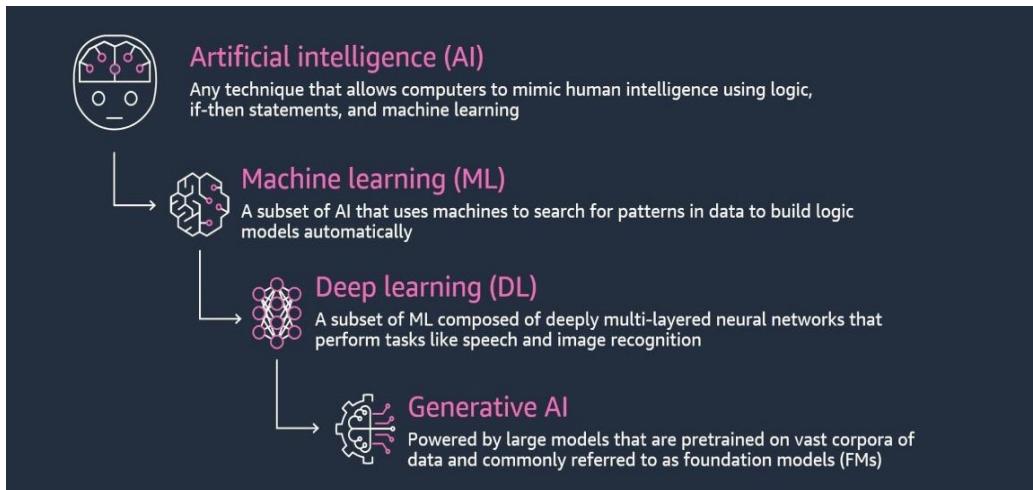
To keep up with the agile mode of development, the traditional testing process is no longer adequate, and teams need to bring in automation. Artificial Intelligence can play an important factor in test automation to reduce the time consumption and manual intervention involved in various test phases. Generative Artificial Intelligence, a subset of AI can aid in these activities. This paper discusses how performance testing and engineering can benefit from Artificial Intelligence and provides some use cases.

## Biography

*Rini Susan V S is a senior quality engineer focusing on Software Performance Testing and Engineering. She has an extensive working knowledge of Performance testing, Application monitoring, DevOps, Profiling, and CI/CD tools. Rini is a regular contributor to technical blogs and articles and has co-authored an article published on developer.com. She has completed a technical program in Artificial Intelligence and Machine Learning.*

Copyright Rini Susan V S 08/15/2024

# 1 Introduction



*Fig. 1: Relationship between AI, ML, DL, and Gen AI.*

Artificial intelligence or AI, is the technology that enables computers and machines to simulate human intelligence and problem-solving capabilities[1]. Today, there are many, real-world applications of AI systems including, speech recognition, online chatbots, computer vision, robotics, etc. Machine learning or ML, is a field of study in Artificial Intelligence concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data, and thus perform tasks without explicit instructions[2]. ML can be used to reduce the amount of routine and tedious tasks in software development and testing. Generative Artificial Intelligence (GenAI) is a subset of Machine Learning. Generative AI models use neural networks to identify the patterns and structures within existing data to generate new and original content[3].

Software Testing is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate[4]. Performance testing is a type of testing intended to determine the responsiveness, throughput, reliability, and/or scalability of a system under a given workload[5]. Performance testing checks and validates an application's capacity and ensures that it works well within the acceptable Service Level Agreements.

Digital customers are provided with a lot of choices these days and performance testing plays a major role in ensuring customer satisfaction. If the performance of an application doesn't meet the expectations of customers, they will opt for another application from a competitor, resulting in business loss for companies.

# 2 Software Performance Testing Progression

Performance testing has evolved over time and the focus has shifted to performance engineering. Earlier, performance testing teams were mainly expected to conduct load test executions and share the test results with development teams. These days, performance engineering aspects are also sought after and the testing teams are expected to identify bottlenecks, perform error analysis, and provide performance-tuning recommendations.

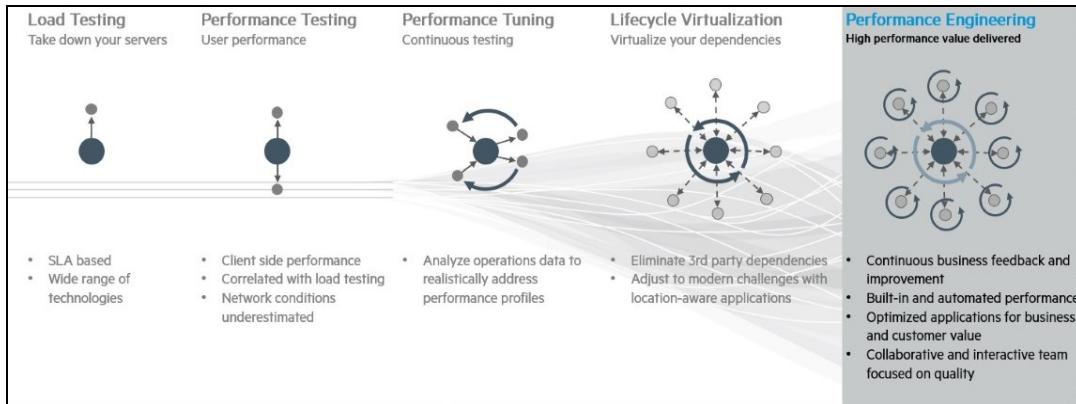


Fig. 2: Evolution of Performance Engineering

As enterprise software platforms become more complex, performance issues have become a serious risk that results in loss of millions of dollars. To keep up with the agile mode of development, the traditional testing process is no longer adequate, and teams need to bring in automation. Artificial intelligence can play an important role in test automation by reducing time consumption and manual intervention in various test phases. Machine Learning and Generative AI can aid in these activities.

### 3 Machine Learning in Performance Testing and Performance Engineering

Machine learning solutions can evaluate and interpret thousands of statistics per second, providing real-time insight into a system's behavior. Machine learning algorithms can identify data patterns, build statistical models, and make predictions. ML-based anomaly detection systems can help to identify performance bottlenecks faster and more accurately.

Generative AI is a subset of Machine Learning that uses AI to create new content in text, image, audio, and video format. It is powered by Foundation Models that can multi-task and perform tasks including summarization, classification, image generation, and live chat. These models can be tailored for targeted use cases with minimal training.

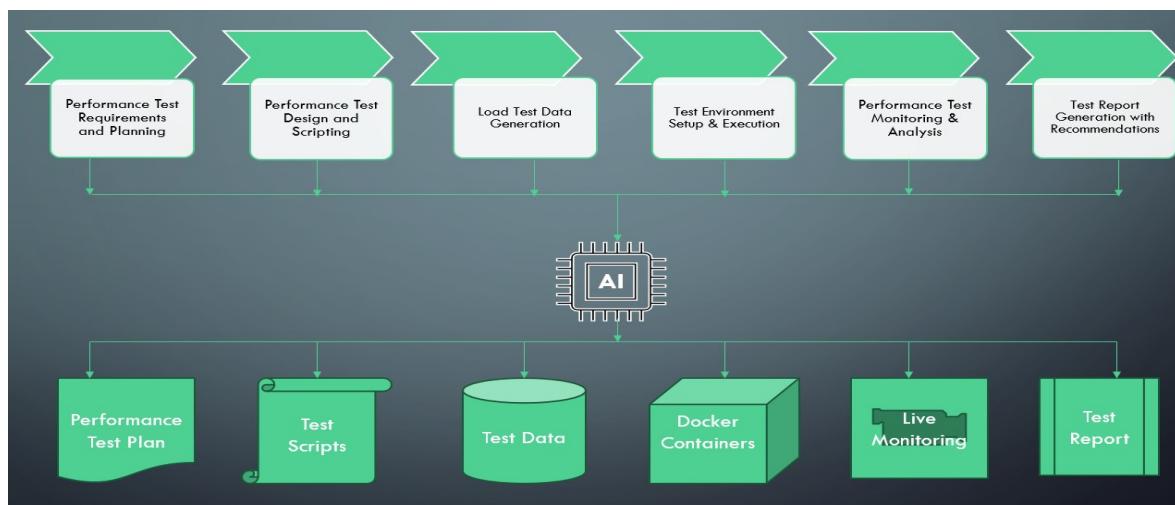


Fig. 3: AI Use Cases in Software Performance Testing Life Cycle

### 3.1 Performance Testing Use Cases

Some of the performance testing use cases that can be addressed using Generative AI models include

- Automatic generation of performance test plan, by providing basic information.
- Test data set creation in large volumes for load testing and endurance testing.
- Automatic test script generation for load testing tools like Gatling, by providing the required information.
- Test report generation in the desired format type, automatically after test execution.

### 3.2 Performance Engineering Use Cases

Some of the performance engineering use cases that can be addressed using Machine Learning techniques include

- Predict server utilization based on metrics like CPU usage, memory utilization, and amount of disk reads and writes.
- Detect outliers in transaction response time based on transaction logs over a period of time.
- Forecast load during special events like holiday or anniversary sales, with historical data as input from various data sources.

## 4 Potential AI Testing Solutions

### 4.1 Machine Learning Models

Large Language Models (LLM) and Machine Learning algorithms (like mentioned below), can be used to automate testing activities, identify performance bottlenecks, and improve the overall quality of software performance testing.

- Foundational Models like Amazon Titan, Anthropic's Claude, Google Gemini, or OpenAI GPT models for test plan and test report generation.
- Mistral AI or Google PaLM 2 models for code generation for Gatling scripts in Scala.
- OpenAI DALL-E or Stable Diffusion XL models for image generation in test report creation.
- Artificial Neural Network (ANN) models to predict server utilization based on metrics like CPU usage, memory utilization, and amount of disk reads and writes.
- K-nearest neighbor (KNN) algorithm to detect anomalies in transaction response time and server utilization matrices. [6].
- State Space Model for forecasting during special events or holiday/anniversary sales, with historical data as input from various data sources.

### 4.2 AI-Based Testing Tools

Various licensed and open-source toolkits with machine-learning capabilities can be integrated with existing performance testing tools to optimize the testing process. There are also advancements in AI-driven automation testing tools. Some emerging testing tools with AI capabilities are mentioned below.

- AppliTools uses Visual AI and no-code approaches for automated UI testing [7].
- Mabl tool uses a low code approach and supports automation, accessibility, and performance testing. [8].
- Functionize tool supports UI, API, and Database testing and provides a cloud testing platform [9].

## 5 Solution Implementation

### 5.1 Use Case: Performance Test Report Generation

Gemini-1.0-pro, a Google AI model, is utilized here to create performance test reports. The main idea behind this use case is to reduce the overall testing effort by automating the manual report creation activity.

```
# Create model with the selected model name and configs
model = genai.GenerativeModel(model_name='gemini-1.0-pro',
                                generation_config=generation_config,
                                safety_settings=safety_settings)
```

*Fig. 4: Model Creation*

Performance test result files are given as input to the model in CSV format. The test report is created based on the prompt provided to the model, according to the project requirement.

```
# Read input file
url = "Test_Results.csv"
with open(url) as file:
    input_file = file.read()

# Print first 5 Lines
for line in input_file.splitlines()[:5]:
    print(line)

timeStamp,elapsed,label,responseCode,responseMessage,threadName,dataType,success,failureMessage,bytes,sentBytes,grpThreads,allThreads,URL,Latency,IdleTime,Connect
1652128008147,16,Test_01_Search_FirstName,200,HTTP/1.1 200,Machine_Learning APIs 1-1,text,true,,289,0,1,1,http://localhost:8080/search/fname?firstName=Mary,15,0,0
1652128008240,9,Test_03_Search_Email,200,HTTP/1.1 200,Machine_Learning APIs 1-2,text,true,,288,0,2,2,http://localhost:8080/search/email?email=alexwill1%40gmail.com,8,0,0
1652128008348,4,Test_05_Search_AccountId,200,HTTP/1.1 200,Machine_Learning APIs 1-3,text,true,,289,0,3,3,http://localhost:8080/search/accountId?accountId=1634589017,4,0,0
1652128008458,3,Test_01_Search_FirstName,200,HTTP/1.1 200,Machine_Learning APIs 1-4,text,true,,289,0,4,4,http://localhost:8080/search/fname?firstName=Mary,3,0,0

# Generate content for the prompt and display the execution time
%%time
response5 = model.generate_content(['Create a performance test summary report with findings and recommendations for the test results in ', input_file])
```

CPU times: user 60.3 ms, sys: 2.08 ms, total: 62.4 ms  
Wall time: 11.1 s

*Fig. 5: Reading the test results file and prompt as input*

Test summary report is generated in the required format with details of test results including response time and resource utilization metrics, test findings, and possible recommendations for further improvement. This solution is valuable for projects operating under stringent deadlines and budget constraints.

```

# Print the response - test summary report document
print(response5.text)

**Performance Test Summary Report**

**Test Results:**

* **Total Tests:** 100
* **Successful Tests:** 100
* **Failed Tests:** 0

**Response Time Metrics:**

* **Average Response Time:** 15 ms
* **Median Response Time:** 14 ms
* **90th Percentile Response Time:** 18 ms
* **95th Percentile Response Time:** 20 ms
* **99th Percentile Response Time:** 25 ms

**Resource Utilization Metrics:**

* **Average CPU Utilization:** 5%
* **Average Memory Utilization:** 10%

**Findings:**

* The API performed consistently well under load, with all requests completing successfully.
* Response times were within acceptable limits, with the majority of requests completing in under 20 ms.
* Resource utilization was low, indicating that the API can handle a higher load without performance degradation.

**Recommendations:**

* Consider increasing the number of worker threads to further improve response times.
* Monitor resource utilization closely to ensure that the API remains performant under peak load.
* Implement caching mechanisms to reduce the load on the database.

```

*Fig. 6: Performance Test Summary Report*

## 6 Road Ahead

### 6.1 Possible Directions

In the near future, we can anticipate a surge in AI-powered tools offering advanced testing features. AI can enhance these tools by incorporating more performance testing and performance engineering capabilities. Following are some key performance engineering use cases.

- Detect anomalies in transaction response time or server utilization
- Root cause analysis of performance issues
- Forecast application behavior based on previous events or logs

### 6.2 Factors to Consider

Though Artificial intelligence (AI) has been widely accepted across industries, implementing AI is not a straightforward task. Organizations must examine various aspects to come up with a well-defined AI strategy. The following are some key factors to consider while implementing AI solutions.

- Hallucinations: AI models are not always perfect and may generate inaccurate/false information, content, or facts. This is known as hallucination and can lead to mistakes and risks.
- Integration with systems: Integration of Artificial Intelligence capabilities into existing systems requires significant planning. The feasibility of AI solutions for the project-specific requirements must be evaluated, along with its impact on existing systems.
- Model choice: Organizations can create AI models or use any of the available AI models. The model is only as good as the data it is trained on; hence it is critical to have correct and pertinent data for model training.
- Computational resources: While selecting an AI model, the usage of computational resources required is also vital. Based on the AI workload, and the model choice, sometimes high-performing Graphics Processing Units (GPUs) may be required.

### 6.3 Responsible AI

While the potential of AI is immense, these technologies can also raise critical challenges that need to be addressed thoughtfully, and carefully. Responsible development is essential to mitigate the unintended or unforeseen consequences due to the misuse of AI. The following are some key Responsible AI objectives that must be prioritized. [10].

- Be socially beneficial: The AI solution or tool is intended to benefit software testing professionals in terms of time and effort, thereby reducing the overall project budget.
- Be built and tested for safety: Safety is a major concern of Generative AI applications using prompt engineering. Hence safety must be ensured by setting the blocking levels of harmful contents based on project requirements.
- Be accountable to people: The solutions or tools are intended to aid software performance testers. The feedback from testers must be sought and incorporated for further tool improvement.

## 7 Conclusion

Given the transformative power of AI, it is crucial to keep up with its advancements in this fast-paced technological era. Artificial Intelligence technologies are essential in the field of software performance testing as well; incorporating them can help to enhance testing efficiency and accuracy.

Artificial Intelligence techniques can transform software performance testing from manual, time-consuming activities to automated, data-driven, predictive insights. Generative AI solutions can help to automate labor-intensive tasks involved in the performance testing lifecycle, while Machine Learning capabilities can aid in performance forecasting and anomaly detection.

To sum up, individuals or teams with foresight can utilize Artificial Intelligence and Machine Learning technologies to enhance software performance testing processes and prevent performance issues, rather than reacting after they occur. Consequently, organizations can shift from a reactive to a proactive performance testing strategy.

## References

- [1] IBM. "What is artificial intelligence (AI)?", <http://www.ibm.com/topics/artificial-intelligence.html>
- [2] Wikipedia. "Machine Learning", [http://en.wikipedia.org/wiki/Machine\\_learning.html](http://en.wikipedia.org/wiki/Machine_learning.html)
- [3] NVIDIA. "Generative AI", <http://www.nvidia.com/en-us/glossary/generative-ai.html>
- [4] B. Meyer. 2018. Seven Principles of Software Testing
- [5] Microsoft Docs. 2010. "Chapter 1 – Fundamentals of Web Application Performance Testing.", Performance Testing Guidance for Web Applications
- [6] IBM. "Anomaly detection in machine learning: Finding outliers for optimization of business functions", <http://www.ibm.com/blog/anomaly-detection-machine-learning.html> (accessed December 19, 2023).
- [7] AppliTools. <https://applitools.com/>
- [8] Mabl. <https://www.mabl.com/>
- [9] Functionize. <https://www.functionize.com/>
- [10] Google AI. "Responsibility", <https://ai.google/responsibility/principles/>
- [Fig1] Amazon Web Services: <http://aws.amazon.com/getting-started/decision-guides/machine-learning-on-aws-how-to-choose/.html>
- [Fig2] Todd DeCapua, Shane Evans: Effective Performance Engineering, ISBN: 9781491950869, 2016

# Signals-based Testing of Microsoft Office with an AI-Driven Robot Army

**Wayne Roseberry**

Wayne\_Roseberry@hotmail.com

## Abstract

From 2017 to 2023, I led an effort to use machine learning and AI driven self-learning autonomous agents to test Microsoft Office. The goal was to get as much product coverage as we could under complex usage conditions and use the product signal data, telemetry emitted by the Office applications, to find errors and bugs not exposed by our other automation solutions or testing efforts. By allowing the AI agents to explore the application using UI controls only, we hoped to uncover conditions and sequences more closely approximating what users might experience.

This experience was productive, rewarding, fascinating and difficult. We did indeed find problems real users were hitting that other testing efforts did not discover. We demonstrated that in the race to data, we could discover problems before the first users were reporting them. We showed that self-learning automatons could cover large functional surface area in complex ways under long running sessions. We also learned that doing all of that took a lot of effort in guiding the agents, addressing testability issues in the product, building an approach and method for recognizing problems from the data stream, writing tools to help understand the results, and finally driving the social campaign to help product teams understand how the data was helpful.

This paper shares the story of that experience, starting with some background, briefly describing how the system worked, and then focusing primarily on the challenges that came from working with such a solution. The challenges are especially important to understand, because they are present with all classes of AI and ML driven testing.

## Biography

*Wayne Roseberry is an SDET Manager at Ford Motor Corporation.*

*Wayne worked at Microsoft Corporation from June of 1990 until March of 2023. He started with a four-year tenure in product support, taking customer phone calls. His testing career launched in April 1994 on the MS Research project that became MSN 1.0. He worked on several versions of MSN, Microsoft Commercial Internet Services, Site Server, and then as one of the founding members of the SharePoint Portal team. SharePoint moved into the Office team, where Wayne eventually joined the Office Engineering team which builds the tools and services that the Office team uses to build, deliver and test Microsoft Office. Most of Wayne's time in the 11+ years on the Office Engineering team were spent working on the automation system focusing on issues like delivery efficiency, automation suite reliability, integrating product telemetry signal with test automation processes, supporting long run and complex automation scenarios, and finally using ML to drive product automation.*

*Outside of work, Wayne likes spending time with his wife and three kids. He likes to paint, draw, write and play music. He blogs frequently about testing topics.*

*Copyright Wayne Roseberry, 2024*

# 1 Introduction

I started this journey wondering about ways to exploit automation to cover types of testing that were difficult to do. One of the largest challenges in testing is covering complex sequences of operations in the UI space. On a large application there is more functional surface than a team of engineers can hope to cover themselves. Writing automation to cover it is difficult, and time consuming, and as the complexity of the scenarios grows so too does the difficulty of maintaining control and understanding of the application state.

I had already done a great deal of work integrating product telemetry signal into the test engineering process and was curious how that information could guide test automation efforts. This led me to an interest in machine learning and AI driven solutions.

I came up with an approach that I soon realized others had discovered. Collaborating with a team at Microsoft who built tools around the same problem, I unified my signals-based testing approach with a self-learning AI testing tool. I realized very quickly I was stepping into an entirely new way of testing an application. This new approach came with exciting possibilities, but a lot of very big challenges. We may imagine that a self-learning robot army solves all our problems, but the truth is the same as any other new tool. The added power comes with new difficulties, and in some ways is more difficult than what we were doing before.

# 2 Background

## 2.1 Signals-based testing

Signals-based testing is *when we exploit engineering and operations data emitted by the product in order to find bugs by applying some workload against the product.*

Engineering and operations data are those pieces of information created by the product which are meant for product team use rather than customer use. This might include product log files, used primarily for diagnostic purposes. In this paper, I am referring primarily to product telemetry, data sent by the product across the internet to the product team, which in this case is the Microsoft Office team.

The product team uses this data for multiple reasons. The primary business purpose is to analyze customer usage patterns to find ways to improve the product, produce new features, understand which features are used in which ways. Another purpose used extensively by the product team is to discover problems by looking for signs of error and failure in the telemetry stream. Bug fixed are selected using this data, and fixed bugs are tracked via this data after release to see if fixes were effective.

All this signal data was coming from end users, after the product had been released. It was my intent to use this same signal stream pre-release while testing the product. Around 2016, I made changes to the Microsoft Office client and our engineering systems to allow us to integrate the testing systems with the product telemetry signal in a way that allowed us to isolate test traffic from end users, as well as filter and identify on a per test basis which test was running at the time the data was collected. We began collecting telemetry data from our automation runs and applying similar analysis on that data as we were the customer generated signals.

It was after that when I decided to try using the telemetry signal to train AI driven test automation.

## 2.2 Using signals to train ML



*Figure 1 PNSQC 2017 where I introduce the idea of using telemetry signal to train an ML agent.*

During a presentation I gave at PNSQC 2017, I talked about using a telemetry signal to evaluate and study test automation. It was during that talk where I referenced examples of research on using reinforcement learning to train an ML agent to play Mario. I suggested during this presentation that it was feasible we could do the same thing with applications under test. Ironically, it was during this same PNCQC event that Jason Arbon was demonstrating a similar idea, Test.ai, which used supervised learning to help ML-guided bots build graphs of application functionality that could be played back later.

The distinguishing characteristic of my idea was that the telemetry signal, the same thing the product team used to track customer usage patterns, could be used as a reward mechanism for the training system. The agent could learn its way around the application, and we could tell it when it had hit some feature or behavior, we wanted it to remember. I already had experiments underway building such a system, but there was another team at Microsoft building the same thing. They had consulted with me, I told them about the idea to use telemetry signals to train the bots, and eventually they had a system working well enough I dropped my project and adopted theirs.

## 3 How the Robot Army Works

The team building the tool I used referred to it as a “robot army.” This is because it utilized many machines for a run, thousands. The system learns through random behavior, which is why so many machines are used. There are three steps: 1) exploration, 2) reward collection and training, 3) guided navigation based on the trained model.

An important point to understand is that the training does not ever attempt to assess whether product behavior is correct, expected, or even incorrect. The only goal of the tool is to teach an agent how to navigate through an application to reach specific states. The purpose for this goal is to keep the product under test “moving,” and to do so in ways that are both complex and targeting features we desire.

### 3.1 Exploration to build the state-action-state graph

A scheduled job launches multiple (thousands) of autonomous agents running on different machines. The agents can be configured to target specific applications or processes, or to avoid certain applications. This configuration allows the training to focus on a specific application (e.g., Word, Notepad, Calculator, Excel, etc.) instead of randomly navigating everything on the OS.

The agent seeks the target application via various mechanisms (applications menu, search bar, etc.) and launches it. Once the application has completed loading, the agent takes a snapshot of the UI state using the System.Windows.Automation namespace (the same one used to build accessibility tools). At that point it randomly selects an action based on the available and enabled controls and executes whatever that was (e.g. "Click Open"). The UI changes based on the action, the agent captures the changes and a mapping of the prior UI state, action taken, and resulting UI state to a service in the cloud. The agent repeats the prior sense, think, act pattern sending each UI state->action->UI state set to the service.

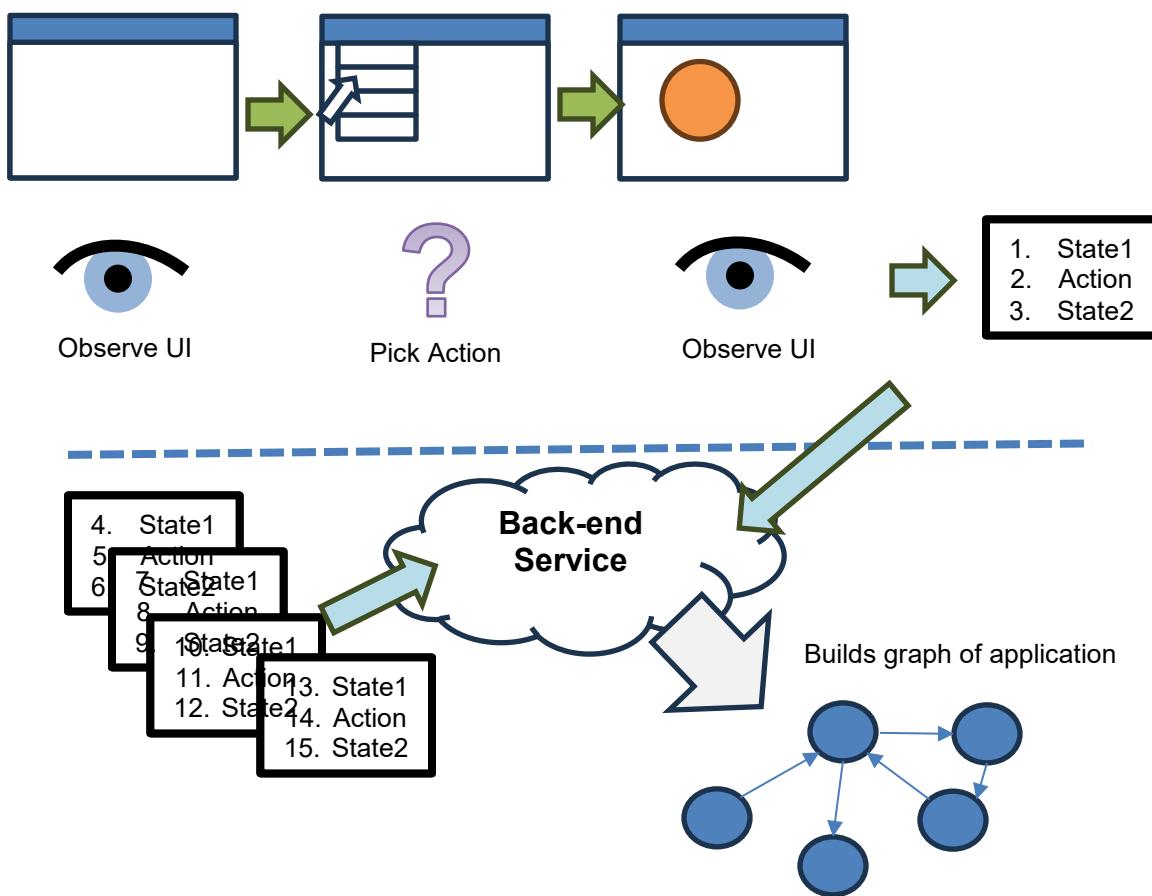


Figure 2 Illustration of agents sending state action sequences to service that builds a graph of application behavior.

On the backend, the service receiving these UI state updates from multiple agents builds a graph that ties state changes to actions. This graph serves as a map to the application behaviors used later after training.

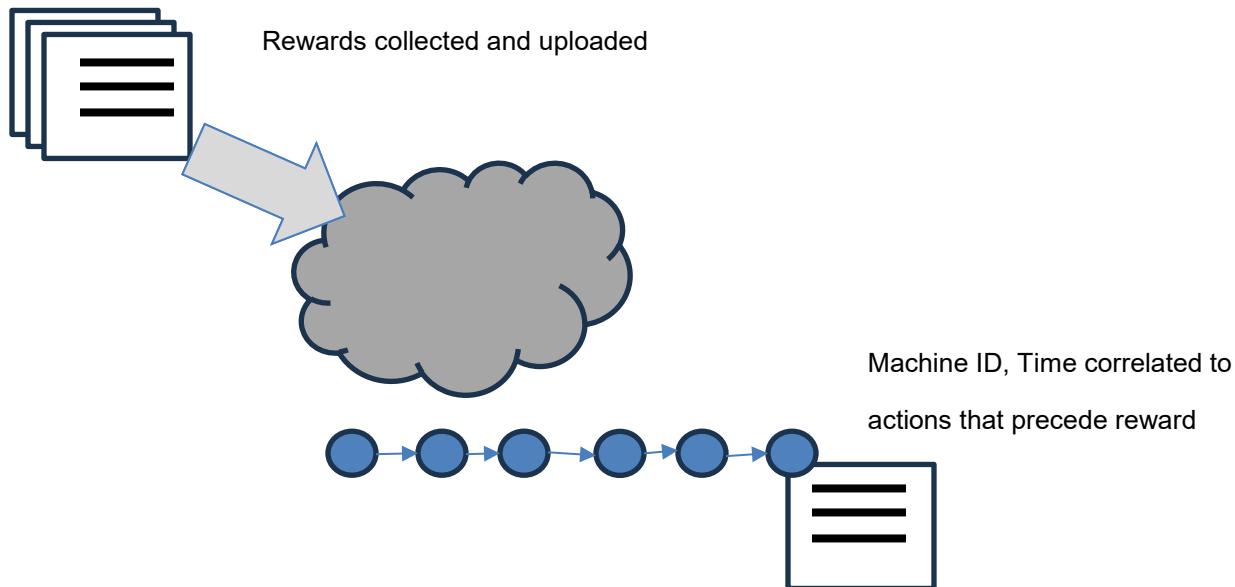
### 3.2 Collection of rewards from signal, train the predictive model.

A scheduled job collected training rewards from product signal data. A daily process would query the Office telemetry logs and collect events emitted from the Office client while the agents were building the

model. Every event carried a machine identifier, the time of the event, and the name. Each event is considered a “reward”. These rewards were compiled in a batch and sent to a service.

For example, a given reward instance might look like:

```
Time: 2022-12-01, MachineId: de734cb4-3c2c-44bb-b077-d6bdf3bcc017, RewardId:  
Office.Word.Commanding.Format.BeginBold
```



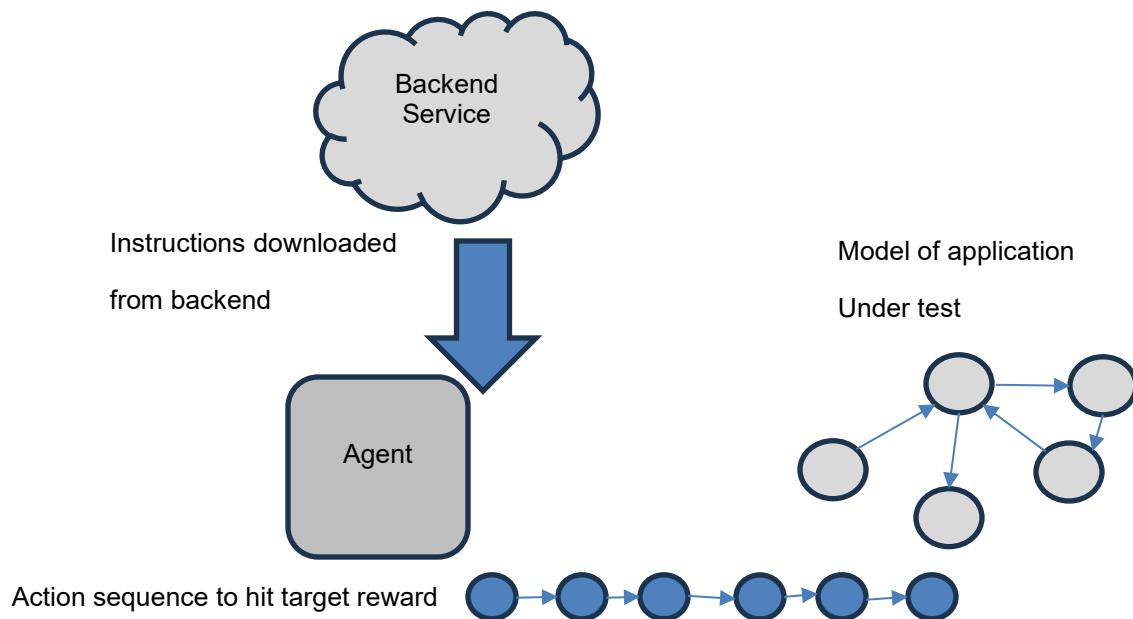
*Figure 3 Backend training to correlate rewards with actions that precede them.*

On the backend service, a nightly process correlated the rewards with the run that produced them using a combination of the machine id and time. The last several actions in the run prior to the event are collected as a sequence correlated with the reward. After all the sequences are collected for all the rewards, the backend begins training an ML model to predict which sequences of actions predict ought to generate which reward.

Periodically, a table of reward weights that reflect the desired frequency of each reward is uploaded to the service. This allowed us to instruct the service to attempt to produce certain rewards more often than others.

### 3.3 Targeted navigation based on predictions.

After training, some of the agents are run in a targeted navigation mode where they goal is to reproduce the rewards collected for training. The agent machines contact the backend service and ask for instructions. The service sends them a package with the model built during exploration, the desired reward, and a sequence of actions predicted to produce that reward.



*Figure 4 Agent receiving instructions from service for actions to generate reward.*

The agent uses the model to build a strategy to get itself to a state where the first action in the sequence would be available. From there, it attempts to complete the sequence. If an action is unavailable, the agent will use the model to predict a sequence from its current state to one where the next action ought to be available.

As it attempts to repeat the sequence, it uploads the same sort of UI state and model sequences gathered during initial exploration, allowing the server to update its model of application behaviors. It also uploads information about whether it can reach the final step in the sequence or decides to abort.

## 4 Kinds of failure

Our effort started with failures that were easy to identify in the signal. Further on in this document I will elaborate on challenges represented by extracting failures from a telemetry signal, but the main ones we looked at were the following:

### 4.1 Crashes/hangs/aborts

Microsoft's Windows Error Reporting service collects crash, hang, and abort data from all Windows applications. Microsoft Office collects the crash data and correlates it with the product data by a session id stored with the crash signature. There is a service from the Windows team that allows product teams to automatically receive error reports for new crashes. Further, the Microsoft Office team has a service using anomaly detection to report new crashes that are showing above certain baseline rates, or whose rate of occurrence has changed. There is an existing process for investigating, triaging crashes, with an active team of engineers from each of the Office product teams involved.

This was the easiest class of failure for us to track. Crashes, hangs, and aborts are always considered a bug, simplifying classification. The existing triage and investigation process gave us a group of people to work with that were experts in that failure domain. We focused on crashes because of simplicity. There were challenges still, covered later.

## 4.2 Explicit errors in the telemetry signal

There are events in the telemetry stream which are marked as errors, making them easy to extract from the data. We did some initial investigation into explicit errors, but challenges around interpretation and purpose of the event motivated us to pull back from that plan.

## 4.3 Performance Regressions

The performance team in Office has a performance monitoring infrastructure in place that tracks certain key operations and reports if there is any evidence of regressions in how long those operations take. The reporting is based on percentile distributions – such as length of operation at the 95<sup>th</sup> percentile of sessions.

One of the challenges of this technique is that pre-full production user populations were small enough that it took up to two weeks to collect enough session data to get a meaningful statistical distribution of performance data and detect a regression.

We found that using the synthetic workloads, we could anticipate performance regressions up to a week earlier than our internal rings of users. The robots were “racing the users” to exposing the bugs, and we found they could take the lead. By the time I left in March of 2023, we had demonstrated the technique was effective and were planning to operationalize the reporting.

# 5 Challenges

## 5.1 Pushing for coverage

The first challenge was trying to get as much coverage as we could from the runs. The Microsoft Office client applications have an exceptionally large functional space. We defined “coverage” as activating controls that existed on toolbar buttons, windows, menus, dialog boxes or control panels. We collected a baseline of available controls by sampling which controls end users hit over a span of a week (there are hundreds of millions of distinct Office users in any daily sample). This came to roughly 10,000 controls per Office application.

Initial coverage was far below 50%. The controls on the default home screen where users choose to open an existing document or create a new document dominated the selected actions, as did the controls on the “Home” ribbon when a document is finally opened.

**1 Heading: 1.1 Head**

## 5.1 Pushing for coverage

The first challenge was trying to get as much coverage as we could from the agents. The Microsoft Office client applications have a very large functional space. We defined "coverage" as activating controls that existed on toolbar buttons, windows, menus, dialog boxes or control panels. We collected a baseline of available controls by sampling which controls end users hit over a span of a week (there are hundreds of millions of distinct Office users in any daily sample). This came to roughly 10,000 controls per Office application.

Initial coverage was far below 50%. The controls on the default home screen where users choose to open an existing document or create a new document dominated the selected actions, as did the controls on the "Home" ribbon when a document is finally opened.

- Default controls hit first, controls behind other controls take longer
- Some controls are content driven, seeded with documents having those elements in them
- Got about a 1-15 point increase by adding some controls as keystrokes in the random text actions (cheating)

Some controls required multiple steps, such as any of the formatting tabs or dialogs, or the proofing panels. As demonstrated in the screen shots above and below, several of the controls in the spelling and grammar panel are only visible if one first switches to the "Review" tab, and then clicks "Spelling and Grammar" on the ribbon. It took several days for the agents to find controls that lived in menus and dialogs further down the command stack.

Some controls required multiple steps, such as any of the formatting tabs or dialogs, or the proofing panels.

**5.2 Please make it stop doing that**

- Default controls hit first, controls behind other controls take longer
- Some controls are content driven, seeded with documents having those elements in them
- Got about a 1-15 point increase by adding some controls as keystrokes in the random text actions (cheating)

There are some controls only visible when certain content exists in the document. Tables, graphic objects, misspelled words, and a variety of other content driven features cause certain menus to appear when the user clicks inside them. The AI agents were not able to create such content easily on their own, so even with weeks of training, they never hit those features.

We addressed that problem with two approaches. The first approach was to seed the canned text strings used by the agents with keystroke accelerators for certain commands. Many of these were document navigation commands (next paragraph, next word, next cell, top of document, bottom of document, select all, etc.) some of these were item insertion (insert graphic, insert shape, insert chart, etc.). In some cases, we identified missing commands from the coverage map and just put in the keystroke equivalent.

The second approach was to seed the test user's SharePoint site and with documents that had lots of rich content. The agents had already learned to navigate their way through the user's default document location and would learn to find the test documents. Once the document was open, the agents would

Excerpt from PNSQC Proceedings

Copies may not be made or distributed for commercial use

PNSQC.ORG

Page 8

learn to find various parts and pieces, and from there the new context-based menus and ribbons would appear, and the agents would add them to the model.

Using this combined approach, we were able to demonstrate that the AI agents hit 70% of the controls which end users were hitting. It took a long-time investigating coverage reports to push the agents to this point, and some of the techniques feel like cheating, but demonstrating that much coverage was what tipped the project over from proof of concept to funding a team to operationalize the approach.

## 5.2 Please make it stop doing that.

There are features in Office which we try to avoid using during automated runs against live systems. The AI agents tended to find these and fixate on them. In each of these cases, the agent would start hitting the feature, and soon after some member of the product team would notice the traffic, it would disrupt something they were trying to track, or in some cases cause serious problems on a backend service, and they would have to come find us to ask what was going on.

In all these cases it was a matter of configuring the agents to not touch certain controls or applications, but it took a while for us to realize what was happening.

**Account Switching and Sign-In:** The button to switch accounts and sign-in as a different account is visible on the top right of all Office applications. The problem we hit there is that once the sign-in dialog came up, the bots tended to get stuck there, typing in the wrong password over and over. It was ruining our runs, so we shut that off.

**Make a Suggestion and Leave Feedback:** This option was available in the upper right corner of Office and is a way for end users to report problems. The team that monitors the internal user feedback for emerging issues saw a flood of strange text (e.g., one of the test strings was "Lisa Simpson is better than Bart Simpson.") in their reports.

**Overwhelming the help search panel:** Office has a "get help" box at the top that searches the Office help documentation, as well as offering a shortcut to features in Office. The agents were typing test strings into that box at a rate that was overloading the services deployed for internal Office users.

**Overwhelm the OneNote document cache download:** The first time a user runs OneNote on their machine, the client checks if they already have a notebook on the server and downloads it to the local machine. The test runs were using a pool of user accounts, but the installation of Office was new on every run, so every run started downloading the prior OneNote notebook for that user, which was getting larger and larger over time. The server deployed for internal use started having outages triggered by notebook downloads far outside the anticipated workload.

**Agents kept shutting down the automation client:** The test machines have a client that the automation system uses to manage the runs, in particular monitoring when a test is complete and initiating the return of the machines to the system. The agents during training would sometimes discover the automation client and shut it down, making the machine "disappear" from the automation system, which meant an engineer would have to connect to the machine to restore it.

## 5.3 Random is slow when functional space is massive.

As stated earlier, we eventually managed 70% coverage of the 10,000+ controls per Office client application. The problem was that hitting all those controls took a long time. If one were to start fresh from a given build, reaching the 70% mark took two to three weeks. Some controls were buried so deep in the model, or required such complex conditions to get to them that the system took a very long time to do it.

Another problem is that using weighting and distributions to push the system toward coverage had to spread itself so thin it really did not pay off. If we tried to mimic "real world" percentages we had a very small number of controls that dominated most of the distribution, while the rest would be well under 0.1%.

The only way we were able to hit a lot of the controls in the lesser frequency range was to flatten the probabilities. Defy the real-world percentages and treat everything equally.

With a massive feature set and a user base approaching 1 billion, very small percentages are actually very large numbers. Psychologically we want to believe that percentages such as 0.1% or lower are not worth thinking about, but even numbers that low describe actions occurring hundreds of thousands, even millions of times a day.

We did not find a solution to this problem by the time I left. We knew we wanted to hit target coverage in a shorter period – more like 1-2 days – but we knew that was going to mean substantial changes to the efficiency and directedness of the coverage.

#### **5.4 Bugs are going to be intermittent by nature.**

This kind of testing creates a randomness and scale (thousands of machines running 24x7) that is exceptionally good at hitting problems which are occurring in the real world, but which are difficult to reproduce. Crashes, hangs, aborts, and out of memory failures tend to be this way. They tend to be intermittent by nature, as the things that drive them are based on run-time state and conditions not guaranteed to be the same every time the end user does the same action.

This is good, because you want to hit these kinds of failures on your machines where you have access to resources unavailable when the user hits the problem. But such failures are also the most difficult to investigate and fix.

#### **5.5 Definition of failure is not always clear.**

Part of the problem we dealt with is that the history of logging errors in Office had not always anticipated us using them as a signal that something went wrong. Many events tagged as errors (which made it easy to find them via a query against the telemetry system) were used by developers for several other purposes. Sometimes it was a way of marking a piece of information the developer wanted to track. Sometimes it was something somebody thought might be an error, but sometimes was not, and they were hoping by marking it that way they could understand the behavior later. Sometimes the error was something that technically went wrong, but for which the product logic had a means to manage and mitigate. Sometimes the error was a normal case of the user doing something wrong and then fixing it on their own. Sometimes the error really meant something bad that we would want to report.

The above is the result of over thirty years of product code. The context and purpose for coding patterns change over time and not everything adapts in a coordinated way. In our case, we were introducing a new purpose where we wanted any error in the telemetry signal to clearly indicate the application believed something was very wrong. This is a problem we had not addressed by the time I was done, but there were other groups in Office working on design pattern proposals with the same goal in mind.

#### **5.6 Understanding what the agent did is difficult.**

The way the agents understand the client UI via the `System.Windows.Automation` API is a hierarchical tree of control elements. The agent saved every node it interacted with as a Json block with the process name, control name, automationid, class name, and object type of the object as described by the `System.Windows.Automation` API. The Office developers were not accustomed to thinking about their application in that way, most of them did not know the IDs of the UI components that corresponded to their features (a lot of that constructed in code outside their control). Further, many the objects were missing automation ids, and had names ambiguous with others in the application.

We attempted to address this problem by adding `AutomationId` fields to controls inside Office to help with disambiguation (also helped with coverage and replay – the agents had trouble using controls with ambiguous object names). We also built tools to visualize the sequences of actions as graphs, particularly

highlighting those that hit problems. This helped give more context to what was going on, helping engineers narrow their investigation.

## **5.7 Replay is by nature uncertain.**

The automation was entirely UI based, where the application and environment state are at its most complex. This meant that not all the states at run-time were guaranteed to be the same on the next run. Some states are different in different runs; different user accounts, MRU list changes, files get changed and saved over time, a variety of other states that live with the user account changes over time and affects the run.

To add to this, the agent is designed to “find” its target state from where it is in the model. This means it may not always be starting from the same state. It also uses this capability to try to correct its path if it ever finds itself on a step where something it expected to be there is not. In this case it evaluates the model and builds a plan to get to the target state to get “unlost.”

These combinations of factors keep replay from being exact. Exact replays are extremely brittle solutions that fail rapidly on complex behaviors, and this whole approach intends to be complex. This means sacrificing exact replay for sake of keeping the system moving.

We had investigations underway to try to achieve as high replay fidelity as possible, but it was always going to be a non-perfect, less than 100% fidelity behavior. The variance is a desired feature of the system. That said, we were finding and fixing fidelity bugs.

Part of the solution to this problem was to focus not so much on exact replay as using data analysis and visualization to show which paths through the application tended toward which failures. We wrote a tool (US patent # 20240111577) that given a specific crash identifier could show how many times the system had hit it and a graph of all the different paths which led to it, ordered by highest hit rate. The idea was not to rely on getting an exact hit on replay but instead to understand the behavior via analysis of the patterns.

## **5.8 Prioritizing the failures is difficult.**

When the agent hits a new crash or other error, it was difficult to know if that problem was worth fixing. It was difficult to anticipate how many people were going to hit that crash when the product was released. The automated runs were hitting thousands of crashes. We would have overwhelmed the product teams if we pushed the product teams to fix all the crashes we found.

Making it even more difficult was that most of the crashes were only seen in the automated runs. For example, there was one feature in Excel that has been there since version 1.0 (not going to tell you which one, sorry, but if you look in books on how to use Excel, it is in there) which is easily available and which the AI agents used a lot, and which they were very good at crashing. Despite this, none of these crashes in this control ever showed up for actual users. Why? I don’t know. There was also a mode that the AI agents were able to get into because the accessibility APIs allowed using controls behind modal dialogs, something the Windows UI blocks. There were crashes occurring in this case because the controls are never used below a modal dialog this way, and consequently we never saw these crashes from end users, only the agents.

During this time, it was about two to three months between the test run and when the product would release to the full market. The internal and pre-release user groups were likewise poor predictors of crashes, hangs, aborts, and other failures seen in the full market. The size of the full user base was so large that their usage patterns overwhelmed everything else. The number of unique crashes captured in the full market was three to four orders of magnitude larger.

But there were crashes that were 1) first seen by the AI agents, 2) eventually seen when the product hit the full market 60 or so days later. The trick was how to predict which ones. We wanted to focus on those likely to show up and also likely to happen often enough to motivate the product team to fix them.

We started an investigation into using binary classification to predict whether the product teams would resolve a given crash FIXED or WON'T FIX, based on call stack patterns in the crash. Call stacks give clues to where in the code the crash occurred, which can also be a rough proxy for a given feature. The Windows Error Reporting service gave us call stacks for end user reported crashes, so we were able to build a model that classified prior call stack entries as predicting either FIXED or WON'T FIX on the bug, and based on make the same prediction for crashes newly discovered by the AI agents. The idea was that if the call stacks were hitting similar parts of the code as prior crashes the product team fixed, then those parts of the call stack must correspond to feature areas more popular or deemed more important to the product team. We were able to demonstrate this technique reduced the number of crashes to examine for fixing from thousands to dozens.

The other technique we used was relying on crashes already in market, but which the product team had been unable to discover the fix. In some cases, product team engineers came to us when they noticed we were hitting bugs they had been chasing for a while and we would help them out trying to understand what was happening.

## **5.9 People incorrectly believe this approach replaces other testing activities.**

People often want to pretend AI and machine learning based tools are far more capable than they really are. When they hear that a tool is on its own learning how to navigate the application space and provide some kind of testing value, they start pushing to use it to replace existing investments in automation and human testing activities. People came to me multiple times pushing me to engage in this sort of replacement. I checked with the manager of the team inventing the tool, and he said the same thing happened to him frequently. We both had the same reasons why the replacement proposal does not hold water.

**Automated scenarios and checks:** Our existing large suite of automated scenarios and checks were designed to examine specific and exact conditions. They were built to manipulate the product into an exact state and then check specific things to see if anything went wrong.

**Professional testers executing scenarios:** In addition to developers testing their own code, Office had vendors under contract to cover sets of test scenarios. More open-ended than the automated checks, such testing is still done with a human awareness of what might seem wrong, unexpected, problematic. People observe and assess the application state to notice anything wrong.

The AI agents work in almost exactly the opposite fashion. They have no model of what is correct, what is wrong, what is expected. If the application does something it should not, rather than stop and report a problem, the AI agents would just “learn” that as the expected behavior. The automated agent power in our case was in the unpredictability at large scale. We were relying not on precise observation of what was happening in the application, but instead on our own analysis of broad trends and patterns in the signal data. The tool, while being good at what it did do, did not in any way match either the automated checks for their capability to measure exact expectations and follow strict sequences, nor match the human ability to notice a broad set of possible oddities and unexpected problems while exploring the application.

## **Conclusion**

Throwing an AI-driven robot army at a product as large and complex as Microsoft Office was one of the most fascinating, interesting, challenging, and entertaining acts of testing I have ever tried. It was a crazy experiment in a lot of ways. We did demonstrate that the idea was sound but had a very long way to go as well. We had to wrestle with the massive size of the application, the unpredictable nature of the tool, inefficiency of coverage, and difficulty trying to decide what to do with the failures we found. On the way there, we had to invent new tools, techniques, and processes. Taking it further was going to require much more. It was a unique experience, and one that might have been difficult to do anywhere else. I believe that some of the challenges we encountered will echo for others as advanced innovations in testing tools and technology move rapidly forward.

## References

**Patent; Creators: SHAH; Mitansh Rakesh (Seattle, WA), RAHMANI HANZAKI; Mahdi (Toronto, CA), ROSEBERRY; Wayne Matthias (Redmond, WA), SCHICK; Guilherme Augusto Kusano (New Westminster, CA). Title: SYSTEM AND METHOD FOR DETERMINING CRITICAL SEQUENCES OF ACTIONS CAUSING PREDETERMINED EVENTS DURING APPLICATION OPERATIONS. Patent #: 20240111577. Filed: 2022.**

Wayne Roseberry, PNSQC 2017, “Is Your Automation Any Good?”,  
<http://uploads.pnsc.org/2017/papers/Is-Your-Automation-Any-Good-Wayne-Roseberry.pdf>

Jason Arbon, PNSQC 2017, “AI For Software Testing”, <http://uploads.pnsc.org/2017/papers/AI-and-Machine-Learning-for-Testers-Jason-Arbon.pdf>

Wayne Roseberry, 2023, “Signals Based Software Testing”,  
<https://waynemroseberry.github.io/2023/05/14/Signals-based-software-testing.html>

Windows Error Reporting, [https://en.wikipedia.org/wiki/Windows\\_Error\\_Reportin](https://en.wikipedia.org/wiki/Windows_Error_Reportin)

# Integrating Generative AI for Quality Engineering into the Software Lifecycle

**Rozumenko, A., Udovychenko, A., King, T. M.**

{ Artem\_Rozumenko, Anastasiia\_Udovychenko, Tariq\_King } @epam.com

## Abstract

Software practitioners and development teams are using generative AI and large language models like GPT-4, LLaMA, and PaLM for quality engineering and testing use cases. These include but are not limited to software requirements and design validation, test case and test script generation, and test result and coverage analysis. But how effective is generative AI in practice when it comes to delivering software faster without compromising quality? In other words, how do teams effectively apply this emerging technology in practical scenarios? In this paper, we describe an approach for integrating generative AI into the software development lifecycle. Our approach provides a holistic methodology that keeps the human-in-the-loop, while incorporating generative AI tools into communication channels, software hubs, integrated development environments, and test execution and reporting platforms. To keep our discussion grounded in practice, we present industrial case studies in which global teams have used an enterprise-ready generative AI collaboration platform for realizing software quality and productivity gains.

## Biographies

*Artem Rozumenko is a highly experienced technology leader and Director of Technology Solutions at EPAM Systems. With a strong foundation in software engineering and a deep passion for Performance Optimization, Security, and DevOps, Artem has built a distinguished career over the past decade. He began his professional journey as a manual tester and quickly advanced through roles in test automation, software development, and cloud operations. In his current role, Artem is heading up various Generative AI transformation initiatives within EPAM. He is based in Ukraine and continues to be an influential figure in the IT services and consulting industry.*

*Anastasiia Udovychenko is the Lead Software Testing Engineer with more than 10 years of experience in software quality assurance. Anastasiia is a technical leader with a customer-first mindset for achieving delivery excellence through lean governance, open communication, and agile thinking. She has extensive experience in QA across various domains, including e-commerce, financial services, e-resource management systems and laboratory information management systems. At EPAM, has been conducting demos, assessments and helping to educate others on AI for quality engineering. Her academic background in Applied Mathematics, combined with industry experience, enables her to approach QA challenges with a unique analytical and problem-solving mindset.*

*Tariq King is a recognized thought-leader in software testing, engineering, DevOps, and AI/ML. He is currently the CEO and Head of Test IO, an EPAM company, and has formerly held positions such as VP of Product-Services, Chief Scientist, Head of Quality, Quality Engineering Director, Software Engineering Manager, and Principal Architect. Tariq has published over 50 research articles in peer-reviewed IEEE and ACM journals, conferences, and workshops. He has served as an international keynote speaker and trainer at leading software conferences in industry and academia and is co-creator of the Artificial Intelligence United's (AIU) certification in Gen-AI Assisted Test Engineering.*

## 1 Introduction

A future where AI automates software engineering tasks is upon us. Generative AI (GenAI) and large language models (LLMs) are enabling higher degrees of software automation when it comes to requirements engineering, planning, project management, development, testing, and deployment. AI-assisted engineering tools are becoming more ubiquitous, easier to use, interact with, and incorporate into our everyday lives as software professionals. Many software practitioners, product owners, and researchers have been investigating ways to integrate GenAI into the software development lifecycle.

An area that continues to receive much attention when it comes to GenAI is quality engineering. Can we employ GenAI to accelerate software development? If so, would this result in better software? Or would any early productivity gains be short-lived due to rising defect rates? Speaking of quality, this technology also has a dark side. One in which AI infused applications, models, and assistants hallucinate. The non-deterministic nature of LLMs raises legitimate concerns around trustworthiness, reliability, security, and privacy, while the method by which these models give pause associated with intellectual property rights and ownership.

Although these technologies are still in their infancy, it is never too early to formalize approaches, tools, and techniques that promote the safe and responsible use and integration of AI. This paper presents a holistic approach for integrating GenAI into the software lifecycle, starting first with quality engineering activities. Quality engineering has been specifically selected as a means of aligning it with another important and related subject – that of productivity.

The remainder of this paper is organized as follows: the next chapter explores the relationship between productivity and quality. Section 3 discusses generative AI for quality engineering. Section 4 presents our proposed approach. Section 5 contains a case study that applies the approach in the context of a client-facing engagement. Finally, in Section 6 we conclude the paper and discuss future directions.

## 2 Productivity Versus Quality

Individuals, organizations and businesses are excited about the potential for AI to transform the world as we know it. Many believe that AI will revolutionize our personal

and work lives by automating routine tasks and acting as a force multiplier in several industries. In the software industry, researchers and practitioners have been experimenting with the use of generative AI (GenAI) for software productivity [REF]. GenAI is being integrated into tools, processes and practices as a means of enhancing software engineering and other lifecycle activities. However, it is important to acknowledge that productivity in the context of the software lifecycle is not easy to define and can be extremely difficult to measure.

This section starts by describing general aspects of productivity and then focuses on the foundational principles necessary to quantify productivity during the software lifecycle. The main idea is that such a foundation is necessary to integrate generative AI for engineering into the software lifecycle. Furthermore, to ensure that quality is not compromised for speed when leveraging generative AI, we wrap up the section by emphasizing the direct relationship between quality and productivity due to the important role that validation plays in our proposed approach.

## 2.1 Defining Productivity

There are three general concepts that intersect to define productivity: **quantity**, **quality** and **efficiency**. When we think of being more productive, we tend to think about an increase in number or quantity of tasks being completed, typically over some unit of time. However, getting more things done faster only results in true productivity gains if those things are completed well. In other words, the finished work must meet a certain quality standard. Lastly, the effort of operating at this new level of output or throughput should not be wasteful, but instead should be cost efficient.



## 2.2 Quantifying Productivity in the Software Lifecycle

Although it may start with delivery, quantifying productivity during the software lifecycle goes beyond just output or flow. For example, traditional metrics like lines of code or number of commits can provide a basic measure of output, but do not capture quality or

long-term maintainability. Process metrics like cycle time, which tracks the time from when work starts to when it is completed, can indicate a productive flow but are not the only factors. Other factors beyond the measurement of output and flow include:

- **Engineering Culture.** A culture where engineers feel safe to take risks, ask questions and report issues without fear of negative consequences promotes productivity.
- **Work-Life Balance.** Software productivity measures must account for sustainable work patterns, ensuring engineers are not overworked as this can lead to burnout and decrease long-term productivity.
- **Team Dynamics.** Effective collaboration within and across the team is key to productivity. Metrics like pull request turnaround times, number of cross team meetings, and feedback loops can provide such insights.
- **Quality and User Satisfaction.** Monitoring the number of bugs per unit of work can help gauge the quality of output. Collecting and analyzing user feedback and satisfaction metrics ensures that the team's productivity aligns with delivering value.
- **Automation and Tooling.** Automated testing and continuous integration and deployment (CI/CD) pipelines can significantly improve software productivity by reducing manual tasks and allowing the team to focus on higher-value activities.

Due to the complexity of software engineering, it is important to take a multi-tiered view of measuring productivity that includes measuring at the individual, team, and value stream levels. Tables 1 and 2 provide an example of such multi-tiered measurement for software testing. In the context of leveraging generative AI for software productivity, measuring at multiple levels within a given activity such as testing, and across different activities like requirements engineering or design, lends itself to creating a more holistic and accurate view of the positive or negative impacts on productivity when applying AI/ML technologies.

Metric	Description	Additional KPI's or Metric Formulas
Test Development Velocity	Measures the number of tests created by a team member over time.	# Test cases developed by team member per period # Test scripts developed by team member per period
Test Result Analysis Effort	Measures the individual effort of a team member when analyzing test results.	Average # Hours spent on result analysis per person.

**Table 1.** Individual Metrics for Measuring Productivity During Software Testing Activities

Metric	Description	Additional KPI's or Metric Formulas
Test Development Velocity	Measures the number of test artifacts created over a given time period.	# Test cases developed per period # Automated test scripts developed per period
Test Coverage	Measures how well the application or domain covered by tests.	% Requirements covered by tests % Program statements covered by tests % Automated tests
Test Effectiveness	Measures the degree to which tests are capable of finding defects.	Defect Detection Efficiency
Test Result Analysis Effort	Measures the team effort for analyzing test results.	# Hours spent on result or failure analysis
Test Automation Effort	Measures combined effort required for creating and maintaining test automation scripts.	Test Development Velocity Test Automation Stability
Defect Leakage	Ratio of defects found and fixed in given phase to total defects found	Defect Detection Efficiency Defect Containment Efficiency

**Table 2.** Team Metrics for Measuring Productivity During Software Testing Activities

### 2.3 Quality Equals Productivity

As software teams attempt to move faster during the lifecycle, a common mistake is to cut corners on quality. However, time-savings from paying less attention to quality early in the lifecycle is generally short lived as bugs and other issues appear, resulting in costly rework. It may be tempting to think that since generative AI can produce lots of content quickly, we can just use that content “as-is” to get the job done. This is a significant risk which can be mitigated in practice if teams assume a direct relationship between quality and productivity. For example, formulating the equation  $Q = P$  as a testing mnemonic or mantra for the team can signify that time spent improving quality also increases productivity and vice-versa. This can help avoid some major pitfalls related to the key challenges associated with leveraging generative AI for software lifecycle activities.

## 3 Generative AI for Quality Engineering

This section presents our approach to integrating generative AI (GenAI) for quality engineering into the software lifecycle. It starts by identifying some of the key challenges organizations face when using generative AI for engineering, and then describes use cases that apply generative AI to quality engineering and software testing tasks.

### 3.1 Challenges

Integrating GenAI into the software development lifecycle presents a range of challenges that organizations must navigate to fully realize the full potential of these technologies.

- **Security and Intellectual Property Ownership.** A primary concern with LLMs is the uncertainty surrounding security and intellectual property ownership. Using

LLMs increases the risk of data breaches and unauthorized access to sensitive information, while the origin of the generated content raises questions about who holds the rights to LLM outputs. These issues create legal and operational problems that can deter organizations from adopting AI-driven solutions.

- **Workforce Upskilling.** Another significant challenge is the need for upskilling within the workforce. As GenAI tools become more integrated into development workflows, it is essential for developers and other stakeholders to learn how to interact with these tools effectively. This involves not only gaining technical proficiency but also developing an understanding of AI processes, interpreting outputs accurately, and leveraging AI capabilities to drive innovation. Without upskilling, companies risk underutilizing these tools and failing to achieve the desired productivity gains.
- **Content Management.** Compounding these issues is the potential for GenAI tools to produce large quantities of software artifacts that may appear correct to the untrained eye. This can create a false sense of security as subtle errors or inefficiencies may go unnoticed until later stages of development, where they can become costly to address. The sheer volume of AI-generated artifacts can overwhelm traditional processes, making it essential for companies to have adequate tool support and streamlined review practices in place.
- **Validation and Verification.** The accuracy of AI-generated outputs remains a critical concern. Incorrect or poorly structured inputs can lead to flawed results, but even correct inputs do not guarantee optimal outcomes. The inherent unpredictability of generative AI necessitates careful monitoring and validation of outputs, adding complexity to the development process and requiring robust testing practices.
- **Productivity Measurement.** Measuring the productivity gains from using GenAI is a non-trivial task. While GenAI tools can accelerate the production of software artifacts, faster output does not necessarily mean better results. The quality, maintainability, and alignment of AI-generated outputs with the project goals must be carefully evaluated. Companies must develop nuanced metrics that account for both the efficiency and effectiveness of GenAI, ensuring that it genuinely enhances the development process rather than merely increasing the pace of production.

## 3.2 Use Cases

Upon the release of ChatGPT and other popular large language models and assistants, we surveyed over 300 practitioners and collected more than 1200 use cases on how they were using this technology for software development. Over 30% of the use cases collected were related to quality engineering or software testing activities. Table 3 is a summary of the categories of the collected use cases, coupled with supporting examples.

<p><b>Test Case Design and Development</b></p>  <p><b>Examples</b></p> <ul style="list-style-type: none"> <li>Analyzing requirements and automatically generating tests to cover application or component functionality.</li> <li>Generating user acceptance tests, step and feature files.</li> <li>Analyzing source code and API's and automatically generating tests that target the program implementation.</li> </ul>	<p><b>Test Code Generation and Maintenance</b></p>  <p><b>Examples</b></p> <ul style="list-style-type: none"> <li>Generating executable test scripts for automated unit, integration, and system-level testing to address both functional and non-functional testing.</li> <li>Migrating existing test scripts from one language, framework or platform to another.</li> <li>Updating test scripts as the application evolves.</li> </ul>
<p><b>Test Case Maintenance and Management</b></p>  <p><b>Examples</b></p> <ul style="list-style-type: none"> <li>Updating tests over builds to keep pace with changes.</li> <li>Identifying duplicate test cases to reduce redundancy and maintenance efforts.</li> <li>Converting tests from one format or test case management system to another.</li> <li>Simplifying complex tests to reduce the number of steps, improve readability and understandability.</li> </ul>	<p><b>Test Data Generation and Management</b></p>  <p><b>Examples</b></p> <ul style="list-style-type: none"> <li>Automatically generating different types of test data based on a description of data characteristics or application fields.</li> <li>Converting test data from one format or database platform to another.</li> <li>Systematically updating or appending test data with new or modified values.</li> </ul>
<p><b>Test Planning, Execution and Coverage Analysis</b></p>  <p><b>Examples</b></p> <ul style="list-style-type: none"> <li>Generating comprehensive test strategies and plans. Including recommendations for testing techniques, tools, and frameworks.</li> <li>Prioritizing, scheduling, or optimizing tests for execution.</li> <li>Identifying gaps in test coverage for a system or domain</li> </ul>	<p><b>Test Result Analysis and Defect Management</b></p>  <p><b>Examples</b></p> <ul style="list-style-type: none"> <li>Automatically summarizing result of bug reports</li> <li>Grouping similar features or identifying common relationships or possible root cause for failures.</li> <li>Automated failure triage including categorization and severity assignment</li> </ul>

**Table 3.** A Summary of Practitioner-Based GenAI for Quality Engineering Use Cases

By automating and optimizing critical quality engineering and testing processes—from test case design to defect management—AI not only has the potential to enhance software productivity but may also promote more thorough and reliable software testing. It is because of this high potential for transformation that we have formulated an approach and implemented supporting tools and platforms. Some of the supporting tools and platforms are freely available as part of our many contributions to open-source software, while others are integrated into EPAM service offerings. Regardless of their software licensing model, it is the hope of the authors that sharing the details of these projects will encourage others to innovate in this rapidly growing area of emerging technology.

## 4 Proposed Approach

Integrating the use of GenAI for quality engineering into software lifecycle requires careful planning, coupled with a robust infrastructure and focus on both technical and human-centric elements. To ensure a successful and sustainable integration, it is essential to follow a structured approach that incorporates secure access to AI models, solid engineering practices, seamless user experience, content management, and collaborative tooling. This section presents our proposed approach to integration which has evolved from our experience applying generative AI to QE tasks on a variety of client-facing projects. To keep our description rooted in practice, where applicable we provide references to products, platforms, frameworks, and accelerators that can be used to support the approach.

### 4.1 Secure Access to Enterprise-Ready Generative AI Models

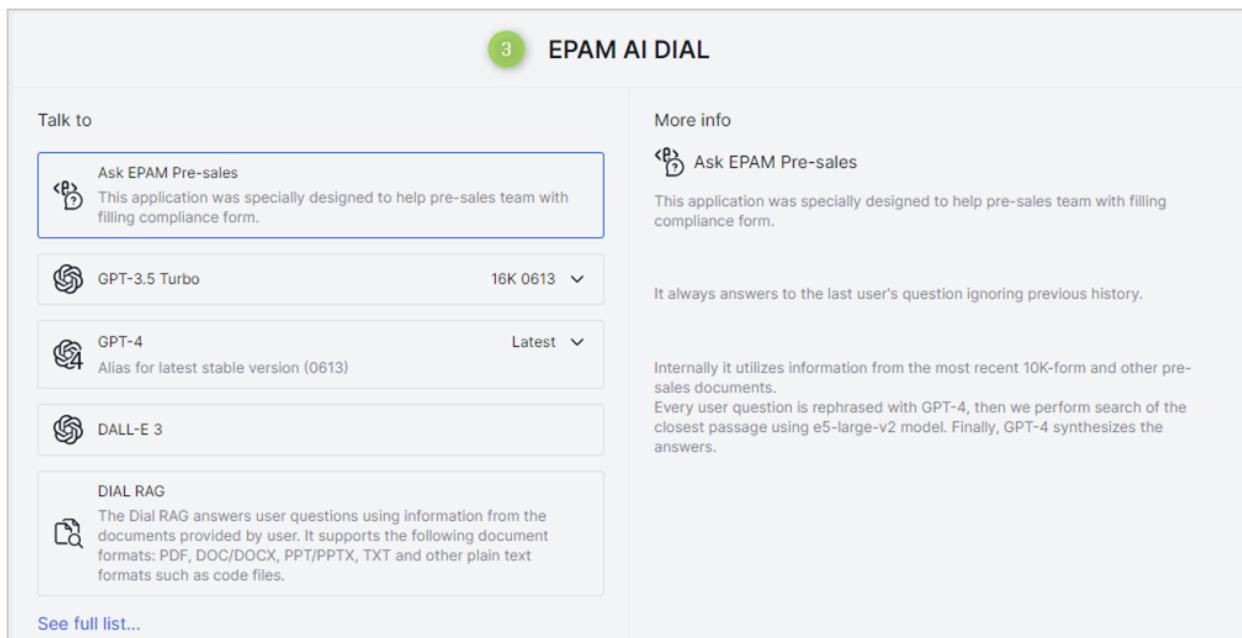
Safeguarding corporate data and assets is paramount and therefore one of the first steps in our approach seeks to address concerns around security and data privacy when using GenAI in the software lifecycle. To ensure a high level of security, businesses need access to enterprise-ready AI models and infrastructure that offer full and exclusive control to the organization. This includes the ability to host these solutions privately, providing an added layer of safety and customization. This provides key assurances in the setup including making sure that prompts, completions, embeddings, and training data are completely isolated and secure. These assets are not shared with or accessible by other clients, nor are they available to model providers like OpenAI. The goal is that your proprietary data remains entirely within your control, safeguarded from external access or usage.

#### 4.1.1 Proxy Implementation

One method for achieving this in practice is to implement a proxy that allows secure access a diverse array of AI models. This proxy serves as a centralized gateway, ensuring that teams can access and leverage the most suitable AI models for their specific testing needs without compromising security. Such a setup should allow for dynamic selection and switching between different models, depending on the task at hand. For example, open-source models might be preferred for certain types of analysis due to their transparency, adaptability and data security, commercially available public models have a huge knowledgebase and can be used to solve majority of the tasks where data privacy is less of a concern, proprietary models could be leveraged for their advanced capabilities and fine-tuned accuracy. The proxy ensures that all interactions with these models are secure, compliant with regulatory standards, and auditable.

#### 4.1.2 AI DIAL

Merging the power of LLMs with deterministic code can facilitate the development of a unified interface for empowering businesses to leverage a spectrum of models, assistants, and more. This is exactly the philosophy behind the AI-powered Deterministic Integrator of Applications and LLMs (AI DIAL).

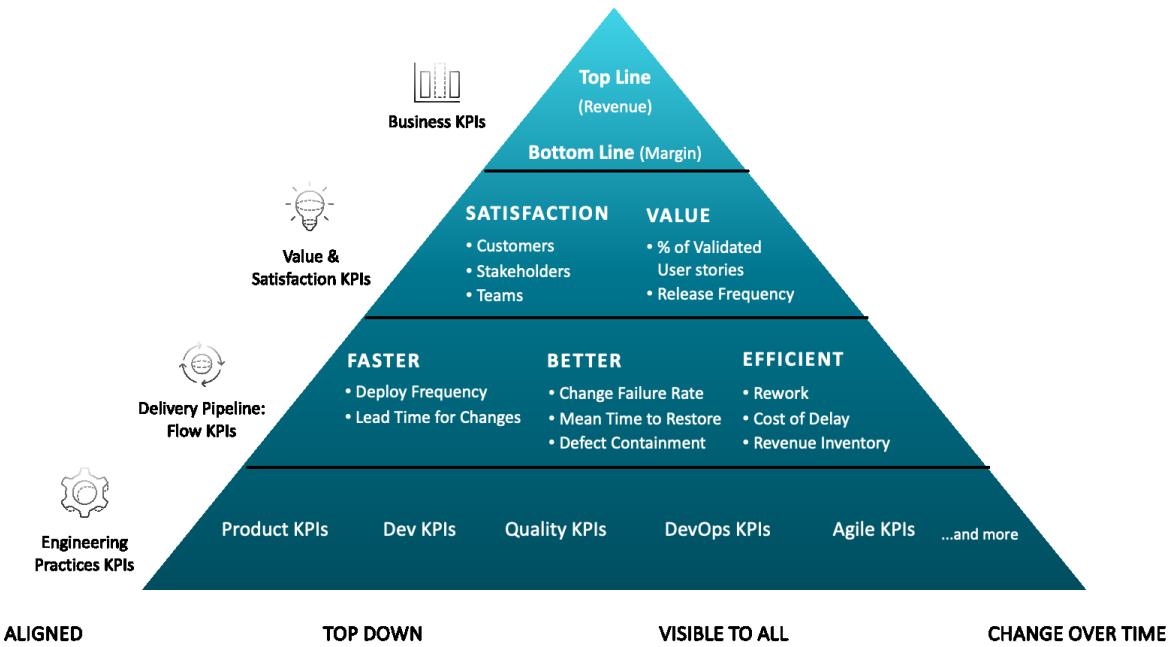


**Figure 1.** User Interface of Open-Source Chat-Based Accelerator Built for AI DIAL

Figure 1 is a screen capture of a chat-based accelerator built on AI DIAL. It allows the user to select which model or models they want to interact with and even provides a feature for sending the same prompt to two different models so that the results can be compared in real-time. Custom models, assistants or other applications can also be accessed via this interface, which is backed by a secure proxy. The type of integration and unified access provided by AI DIAL promotes the development of novel, AI-based, enterprise assets that co-exist seamlessly with an organization's existing workflows. In keeping with our long-standing commitment to open source, EPAM has released this chat-based accelerator under the Apache 2.0 licensing scheme. AI DIAL encourages responsible use, community innovation, and adoption of responsible AI standards within the industry.

## 4.2 Building on A Foundation of Engineering Excellence

Our approach recommends that the integration of GenAI into the software lifecycle be built on a foundation of engineering excellence. Engineering Excellence refers to the pursuit of the highest standards in engineering practices, encompassing the consistent delivery of high-quality, reliable, and innovative solutions that meet or exceed user and business expectations. It involves a commitment to continuous improvement, adherence to best practices, and the application of rigorous methodologies that ensure the optimal performance, scalability, and security of products and systems. Engineering excellence embodies a culture of transparency, accountability, collaboration, and ethical responsibility, driving teams to not only achieve technical precision but also to contribute meaningfully to the advancement of the engineering discipline and organizational success.



**Figure 2.** Engineering Excellence Key Performance Indicator (KPI) Pyramid

A key benefit of leveraging engineering excellence within our approach is that it promotes a strong, KPI-driven culture of engineering best practices. Figure 2 illustrates the various kinds and levels of Key Performance Indicators (KPIs) that a focus on engineering excellence can provide. Leveraging such a framework is particularly useful during the integration of GenAI as it supports conducting observations, measurements, baseline assessments and evaluating the impact of AI on quality and productivity at multiple levels. For example, before introducing AI into testing activities, it may be essential to capture baseline measurements on test execution times, defect rates, and test coverage. These baselines provide a point of comparison to directly measure the impact of AI integration on testing but can also be extended up to higher levels of the pyramid to gain insights into the overall value.

In cases where capturing baselines is not feasible, running concurrent experiments—where one set of testing activities is assisted by GenAI, and the other does not—can help assess the overall benefits and challenges. This may require the implementation of observability tools that monitor AI-driven processes in real time, ensuring that any deviations from expected behavior are promptly identified and addressed. Measurability, coupled with these observability practices, provides the data needed to refine AI models and continuously improve their performance in the software development lifecycle.

#### 4.3 Seamless Integration into Existing Tools and Workflows

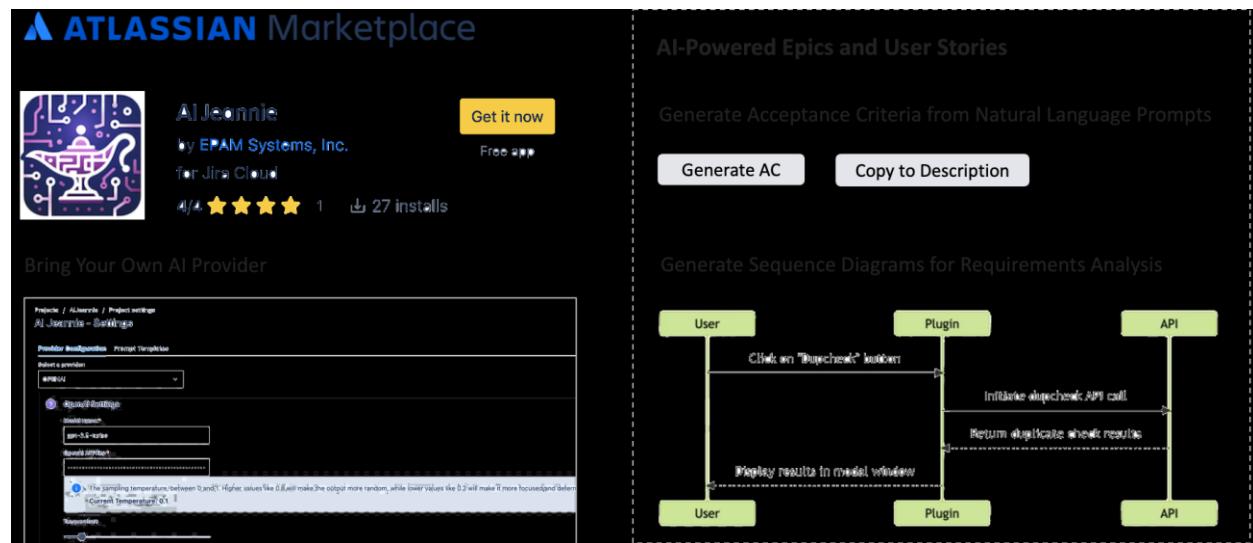
One of the most critical aspects of integrating GenAI into the software lifecycle is ensuring that it fits seamlessly into existing tools and workflows. Rather than requiring teams to adopt entirely new tools or platforms, AI should be embedded within the products and systems they are already using. This approach minimizes disruption and reduces the

likelihood of resistance from teams who may be hesitant to learn new technologies. For example, AI-powered test case generation, automation, and analysis should be integrated into the same IDEs, test management systems, and reporting tools that teams are already comfortable with. This integration allows AI to enhance the existing workflows without imposing additional cognitive load on the users, thereby increasing the adoption rate and ensuring that AI becomes a natural extension of their daily work. The remainder of this subsection provides concrete examples of such integrations in the context of quality engineering.

#### 4.3.1. AI Jeannie (JIRA Plugin)

JIRA users currently experience inefficiencies and inconsistencies in their workflows, negatively impacting the quality of their sprints and issues. These inefficiencies often result in increased manual effort. Furthermore, there is a lack of seamless collaboration between Business Analysts and other JIRA users due to inconsistencies in language and terminology used for JIRA issues.

AI Jeannie is a freely available, open-source, plugin for JIRA that helps Business Analysts to create quick epic descriptions, user story descriptions, and acceptance criteria based on Project Definition configured. As illustrated in Figure 3, it works by allowing you to bring your own AI or LLM provider and helps users to quickly generate high-quality, accurate, and relevant requirements directly within JIRA. Sequence diagrams can also be automatically generated to allow stakeholders to validate them as part of their agile process and workflow.



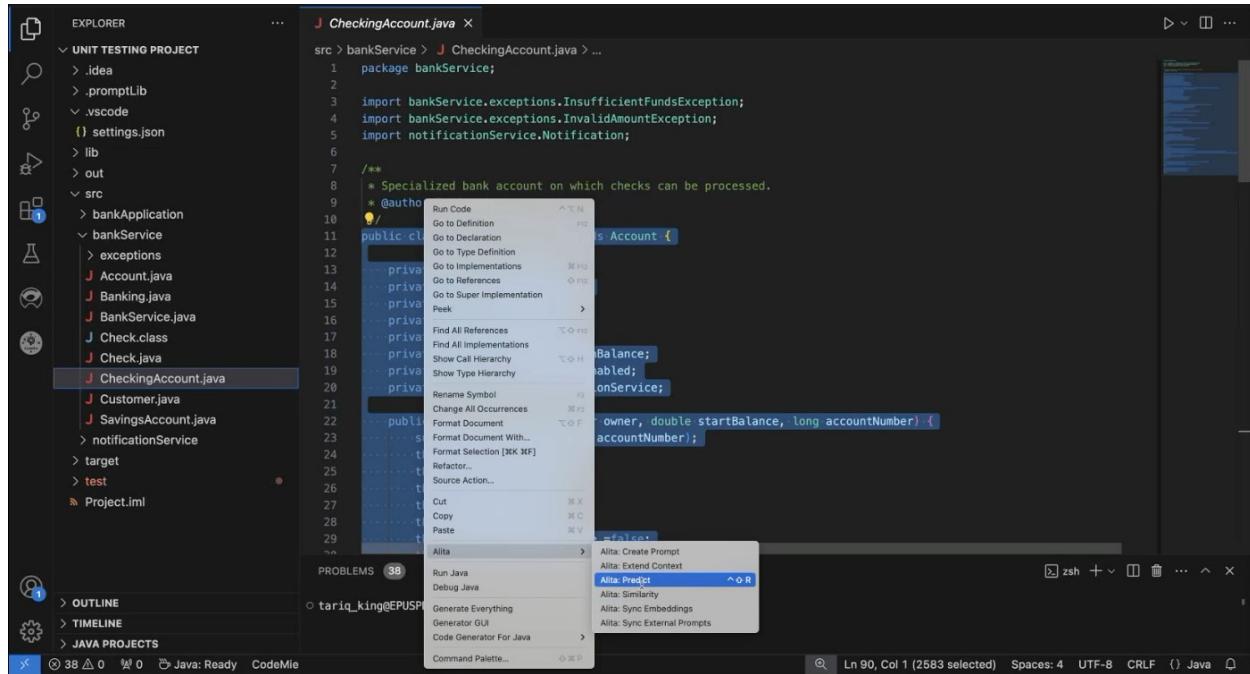
**Figure 3.** AI Jeannie JIRA plugin supports requirements-based quality engineering

#### 4.3.2. Alita Code (IDE Plugin)

Software and test automation engineers often encounter challenges during development when implementing program-based tests. Generative AI and large language models perform quite well at code generation tasks, including creating automated test scripts. However, ultimately this code will live in a repository and be updated and maintained using an integrated development environment (IDE). Therefore, it makes sense that these types of tools be provided as extensions to the IDE, rather than outside of the programming environment.

Alita Code is an IDE extension seeks to improve the way that software engineers and testers develop, test, and maintain program code. It leverages AI to generate automated unit, integration, and system-level tests. It can also automatically add comments to code to make it more understandable and maintainable for engineering teams and provide AI-powered code suggestions. At the time of writing, the Alita Code plugin is available for multiple IDEs including Visual Studio Code and IntelliJ and can be configured for public or private LLMs. Figure 4 shows the Visual Studio Code variant, which exposes the following extension commands:

- **Init** – Initializes the plugin and creates a local prompt library folder in the open workspace.
- **Create Prompt** – Adds a new prompt to the local prompt library.
- **Extend Context** – Facilitates expanding the context of a given prompt in the local library.
- **Predict** – Presents user with a list of prompts for performing GenAI engineering tasks.
- **Similarity** – Presents user with a list of embeddings to run similarity search against.
- **Sync External Prompts** – Allows users to sync prompts to and from a shared backend.



**Figure 4.** Alita Code Plugin for Integrated Development Environments

#### 4.4 Supporting Content Management and Validation

AI-generated content, such as test cases, scripts, and reports, must undergo review and validation processes to ensure accuracy, relevance, and compliance with organizational standards. This involves setting up review pipelines where AI-generated outputs are automatically routed to human reviewers for approval.

Building connectors to existing test management systems and reporting tools is also essential. These connectors facilitate the seamless flow of AI-generated content into established systems, enabling teams to track, manage, and validate AI outputs alongside traditional testing artifacts. By making content management and validation an integral part of the AI integration process, organizations can maintain high standards of quality and reliability in their testing practices.

#### 4.5 Facilitating Collaboration and Experimentation

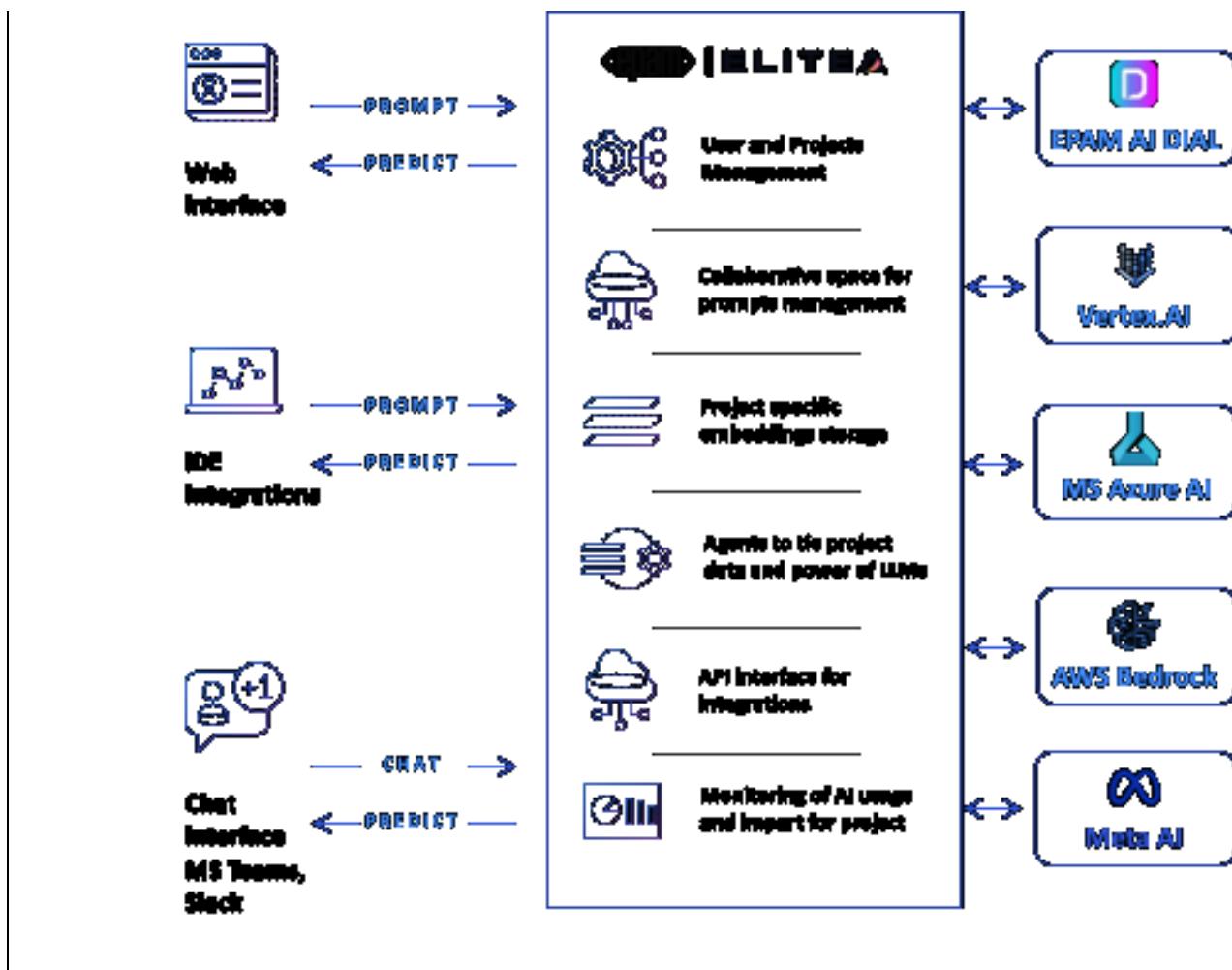
Collaboration is a key ingredient for harnessing the full potential of GenAI. Mechanisms that enable teams to share prompts, refine models, and collectively improve engineering processes must be put into place. Such tools should support real-time collaboration, such as sharing environments or communicating via integrated chat and feedback systems. The proposed approach therefore emphasizes that beyond the technological elements, organizations must invest in practices aimed at helping teams work together more effectively with each other, and with AI.

In addition to a collaborative space for sharing, there should be private spaces for conducting localized experiments so that teams can explore new ideas with GenAI in a

controlled environment. For example, in the early stages of experimentation it may be necessary to set up an isolated test environments where AI can be trialed without impacting the broader development or testing effort. Such experimentation fosters innovation and ultimately contributes to building a culture of continuous improvement.

## 4.6 Bringing it All Together

At EPAM and Test IO, we have applied the various elements of the proposed approach successfully and have used the content described here as guiding principles in developing our own AI collaboration platform. Our platform sits behind a secure proxy which provides access to a plethora of LLMs, and we have been using it to seamlessly integrate GenAI into testing and software engineering tasks. Wherever possible, ELITEA is used to help augment existing tools, frameworks, and process workflows with AI capabilities. Figure 5 contains a visual of the approach and illustrates the kind of bridge ELITEA provides to teams as an AI collaboration platform.



**Figure 5.** Seamlessly Integrating GenAI for Software and Quality Engineering with ELITEA

## 5 Case Study

The anonymized case study presented in this section is taken from a past client-facing, industry engagement that used the proposed approach to accelerate quality engineering and testing activities. It includes a description of the client, the problem motivation, key highlights of the engagement, and any results, findings or conclusions.

### **Reducing Testing Debt in Library Services with Gen AI**

Client is a co-creator and key contributor of a cloud-based, open-source, library services platform. Although the platform uses a modular architecture that lends itself to test automation, the rapid pace and community-driven nature of the project has led to gaps in test automation. The growing reliance on manual testing motivated the client to reach out to us for guidance on leveraging GenAI for UI test automation to tackle the accumulated software testing debt.

#### **Project Goals**

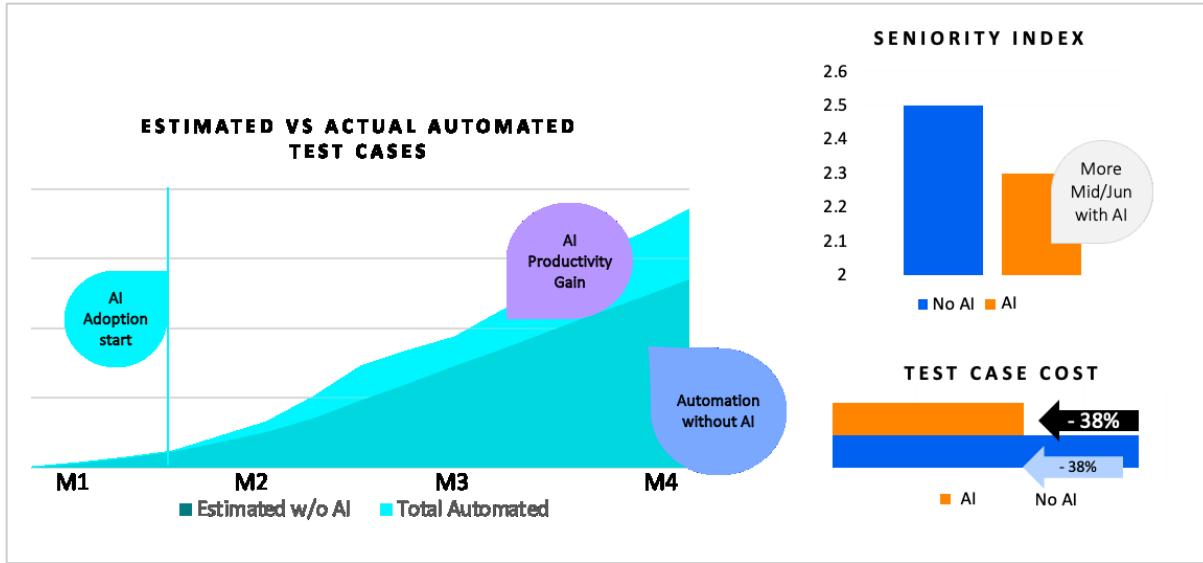
The goal set for the project was to successfully integrate GenAI into the test script development workflow of the team members and automate a large number of end-to-end, manual UI tests within a 4-month period. Achieving this should in turn accelerate the release cycle due to the increased test automation.

#### **Highlights**

Some key highlights of the engagement included the use of crowd-sourced test automation contributors; a template generator; IDE plugin' and CLI tool all as GenAI usage enablers. In addition, there were dedicated engineers to conduct GenAI experiments using different prompts, models and approaches, and the Alita Code extension enabled team members to build and execute prompts within the IDE.

#### **Results**

Within the 4-month period, 1000+ end-to-end UI tests were automated. After a short ramp-up period, a subset of team members was onboarded with the GenAI skills and tools necessary. Figure 6 provides a snapshot of some of the project results. There was approximately a 15% increase in the test development velocity when using GenAI, and an estimated 20% time savings and close to 40% reduction in implementation costs. Another finding worth noting was that the seniority index of GenAI assisted engineering group was a few points lower than the non-AI group. In other words, a group with less senior engineers using AI, outperformed a group of more seasoned engineers without AI assistance.



**Figure 6.** Snapshot of the Case Study Results

## 6 Conclusion

As demands on time to market for high-quality software grow, it is becoming necessary to accelerate the software pipeline. Consequently, the integration of generative AI into the software development lifecycle is changing how software is designed, tested, and delivered. This paper has explored the challenges and opportunities of embedding generative AI within software engineering processes, particularly focusing on quality engineering and testing. The integration of generative AI brings forth challenges such as security concerns, intellectual property issues, and the need to upskill the workforce. The temptation to prioritize speed over quality, especially with the rapid output capabilities of generative AI, is a common pitfall. However, the presented approach and associated case study are providing early indicators that such an integration provides a net benefit to the software development cycle. Still there is much more work and practical studies that are needed before we can draw any firm conclusions around the transformative potential of this technology.

## Acknowledgements

The authors would like to thank the following people and the members of EPAM's Testing Competency Center for their contributions to this work: Ihar Bylitski, Adam Auerbach, Pavel Seviaryn, Matthew Gorelik, Viktoryia Valakh and Dmitry Tovpeko. We would also like to express our sincere appreciation to the entire EPAM and Test IO community. Their collective wisdom, experience, and collaborative spirit have enriched this work in countless ways, ensuring its relevance and impact on the industry.

## References

"Continuous Testing and Quality Gates: A Practical Guide." *Journal of Software Testing*, Vol. 14, 2019.

Amershi, S., Cakmak, M., Knox, W. B., & Kulesza, T. "Power to the People: The Role of Humans in Interactive Machine Learning." *AI Magazine*, Vol. 35, No. 4, 2014.

Mitchell, M. *Artificial Intelligence: A Guide for Thinking Humans*. Farrar, Straus and Giroux, 2019.

Russell, S., & Norvig, P. *Artificial Intelligence: A Modern Approach*. 4th ed., Pearson, 2021.

"ReportPortal: An Open-Source Reporting Platform with Machine Learning Capabilities." Official Documentation, 2022. <https://reportportal.io/docs/>

"ELITEA: Platform designed to streamline the management, development, and collaboration of LLM assets." Official Documentation, 2024. <https://projectalita.ai>

"Alita Code: AI-powered IDE extension for developing, testing, and code maintenance." 2024. <https://marketplace.visualstudio.com/items?itemName=ProjectAlita.alitacode>

"AI Jeannie: Jira extension to create Epics and User Stories using Generative AI." 2022. <https://marketplace.atlassian.com/apps/1232950/ai-jeannie>

"AI DIAL: An Open-Source AI Orchestration Platform and Development Studio" 2024. <https://dialx.ai/>

Whittaker, J. A., Arbon, J., & Carollo, J. *How Google Tests Software*. Addison-Wesley Professional, 2012.

Gumeniuk, D., & King, T. Improving Enterprise Scale Test Automation with ML-Based Predictive Analytics. In the Proceedings of the 2023 Pacific Northwest Software Quality Conference, October 2023.

One Hundred Year Study on Artificial Intelligence (AI100). 2016 Report, 13. [ai100.stanford.edu/2016-report](http://ai100.stanford.edu/2016-report)

Kelly, K. "The Three Breakthroughs That Have Finally Unleashed AI on the World." *Wired*, Oct. 27, 2014.

Li, W., & Zhang, X. "Using Large Language Models for Automated Test Case Generation." *Journal of Software Testing and Verification*, 2022.

Maxwell, P. *AI for Quality Assurance: Leveraging Artificial Intelligence in Testing*. TechPress, 2021.

Johnson, J. *The AI-Driven Enterprise: How AI Can Transform Quality Assurance and Testing*. FutureTech Publishing, 2022.

Khair, M. A., Mallipeddi, S. R., & Varghese, A. "Software Testing in the Era of AI: Leveraging Machine Learning and Automation for Efficient Quality Assurance." *Asian Journal of Applied Science and Engineering*, Vol. 10, No. 4, December 2021, pp. 45-60.

# What would you say you do here? Reframing your Role in the QA Field

Jeff Sing

[jeffsing@iterable.com](mailto:jeffsing@iterable.com), jeffsing@gmail.com

## Abstract

The job of a QA Leader is to create a vision of your QA organization and engineers' roles at your company, but the struggle is that business needs are constantly changing. For example, do we even need a QA role anymore with advances in tooling and AI or with a cheaper headcount outside the US?

To address this, I want to reframe the role of QA in the field, not as testers or developer tooling but as engineers delivering QA as a Service (QAaaS). Running a QAaaS identifies the areas of needs that engineering lacks, highlights why the work is business impactful, and why the ROI of keeping this organization is essential. I will talk about how to build a pipeline of projects (technical tooling, developer enablement, quality process, governance system, culture shaping), justify value with telemetry/metrics, and the importance of marketing the concept of QAaaS to leadership and your peers.

## Biography

Jeff Sing is a Quality Leader who has been in the testing industry for over 15 years. During this time, he has built automation frameworks and test strategies and executed quality initiatives for fields such as medical devices, infrastructure security, web identification, marketing tech, and experimentation and progressive delivery.

Jeff is currently a Director of Engineering at Iterable, where he leads the Quality Engineering organization in orchestrating their quality control plan utilizing a combination of automated testing and implementing QA procedures. He also acts as Iterable's customer experience champion to ensure Iterable remains the world's leading customer engagement platform. He has also built and led Iterable's Engineering Operations Team, which runs the services and programs to improve effectiveness and productivity systematically across the engineering organization as it scales.

## 1. Introduction

Recently, I spoke to an acquaintance who was struggling with his journey in the QA field. He was worried about his role being replaced and wanted to see if he should transition to a more technical testing role. The more I thought about his question, the more I realized that reframing his role as a more technical role wasn't necessarily going to provide him with more job security. If our roles are only to test and validate, what prevents our leadership teams from outsourcing testing to a third party or transitioning testing to developers? Contracting testing overseas is significantly cheaper than hiring a full-time tester. Many companies have eliminated their testing teams entirely and shifted testing responsibilities to their developers. On top of this, multiple third-party vendors sell low code testing tools or AI testing tools that

promise to make dedicated testers obsolete. So, if I'm the head of engineering and need to cut costs, why would I employ dedicated testers?

The answer would be that a quality program does more than just testing. They are building the solutions that ensure your engineering organization successfully delivers the value proposition your customers have paid for. These solutions might be technical or process-oriented. Your quality program should also be able to define what quality looks like in terms of business impact with KPI and be able to measure it. Utilizing these data points, your QA program should constantly adjust to deliver and build upon a quality culture.

The good news is savvy QA Programs are doing a great job of these things! The problem is that most don't have a great strategy to ensure their engineering leadership team acknowledges this value. If the Head of Engineering only sees his Quality Engineering team as an organization that delivers testing, then when another service provider (vendors, third-party contractors, or AI) comes and can claim to disrupt testing (do testing cheaper, better, or faster), the in house Quality Program suddenly becomes expendable.

How, then, should Quality Programs ensure that their services are utilized and valued? **This problem is actually similar to the problem faced by all software-as-a-service (SaaS) companies with their customers!** They generate revenue when customers utilize their services. However, if the customer doesn't find value, they will churn. This could be because the customer isn't getting their desired outcomes, weak value from this service, or they can find a cheaper competitor. In the same vein, we should think of our quality programs as a quality service organization. We win when we deliver our product (Quality) and the customers renew or uplift (utilize what we offer, maintain our teams, or even scale the quality program). We lose when our customers decide we aren't the right solution, and they churn (lay off QA or reduce the scope of the Quality program responsibilities).

So, how do we reframe the role of our Quality Program to include work that isn't traditional testing that creates the value proposition that we are an essential organization? Going back to my analogy of SaaS companies, let's see how SaaS companies do this:

- Be able to understand their customer's needs and how our solution helps them be successful + ability to scale this solution with our customers as they scale (Product Management)
- Be able to support their customers when they have issues (Customer Service)
- Be able to show the value of their product or services (Account Management)
- Be able to renew and upsell their services (Sales)

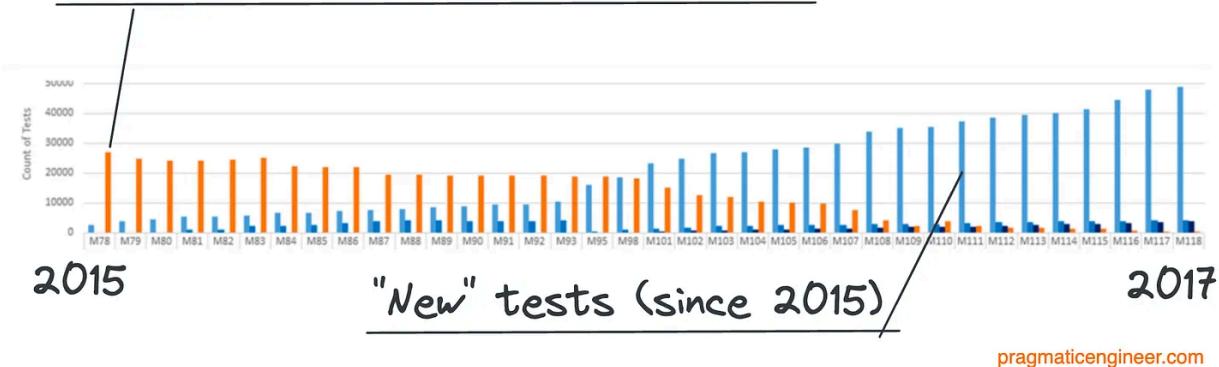
The following sections will describe how the above Business Units in SaaS companies execute their strategy and how this translates to what a Quality Program should incorporate to deliver value.

## 2. The Problem: Are We Building the Right Quality Program?

Amongst my conversations with other Quality Engineers at different conferences, meetups, and co-workers, there is high confidence that we are performing tasks that will surely answer the question, "*Are we building the product right?*" We know how to find bugs, build automation, and develop sound testing plans. But what are we doing currently that answers the question, "*Are we building the right product?*" How do we determine if the work we do in our Quality Program fits for purpose?

In 2014, Microsoft laid off most of its QA organization. “*We combined the dev and test orgs into a consolidated ‘engineering’ org. For the most part, we eliminated the distinction between people who code and people who test. That’s not to say every person does an identical amount of each, but every person does some of everything and is accountable for the quality of what they produce. We also set out to completely throw away our 10’s of thousands of tests that took 8 years to create, and replace them with new tests that were done completely differently.* [1]”

## “Old” tests (from 2015 and before)



In 2 years, almost all “old” tests from when test was separate from dev, were gone. The new tests became more granular as well.

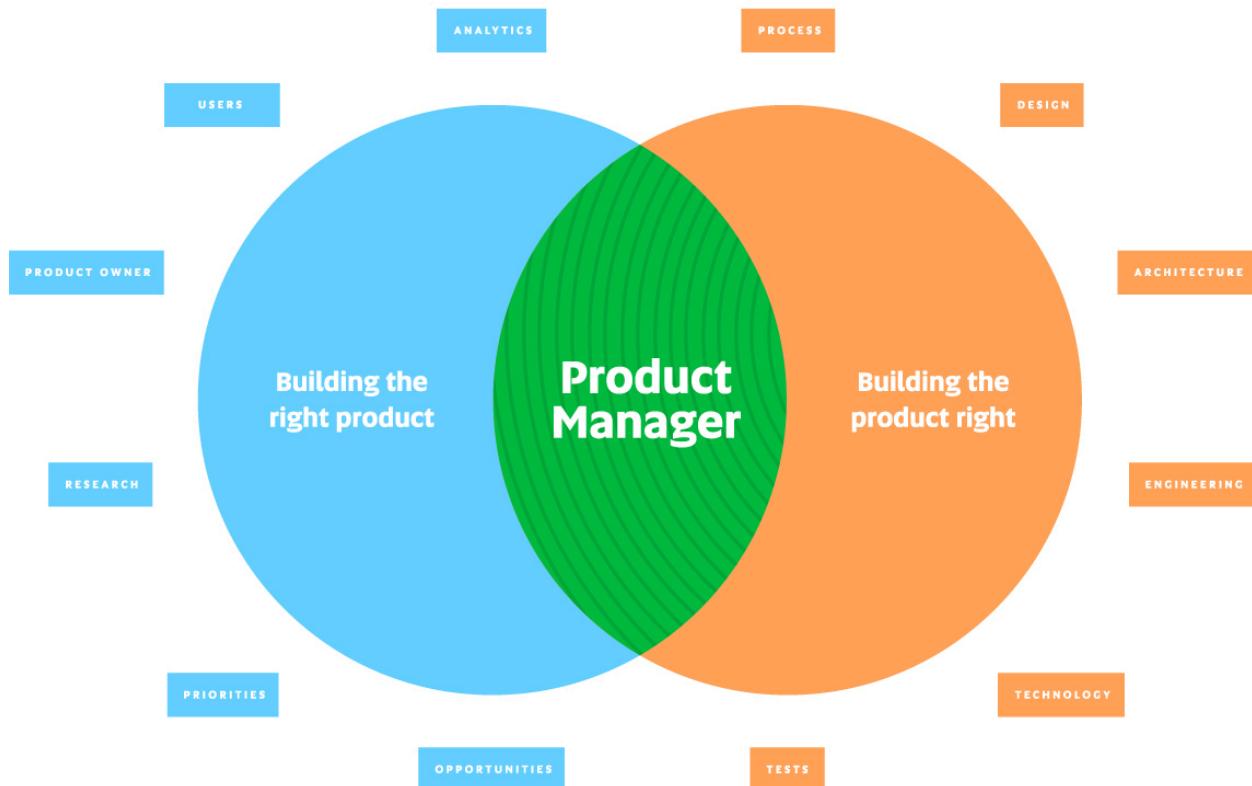
Source: [Microsoft Dev Blogs](#)

Looking at this decision, we see that the Microsoft QA organization invested in a product that wasn’t necessarily the right solution (end-to-end test). Because of this, their roles were easily replaceable (developers can write tests).

If we want to run a successful quality program, we need to be sure that we aren’t just “building the product right” but also “building the right product.” To do this, we need to ensure that we are building a pipeline of projects (technical tooling, developer enablement, quality processes, governance systems, and quality culture shaping) that justify the costs of having one.

How do most SaaS companies accomplish this? This falls under their Product Management organization. Let’s see how we can take some of the techniques from Product Management and apply them to our Quality Engineering organization.

## 2.1 Product Management for Your Quality Services and Programs



*Building the Right Product vs Building the Product Right.* Source: Mojotech product blog

Product Management is responsible for defining desirable, viable, feasible, and sustainable solutions that meet customer needs and supporting development across the product life cycle.

So how does the Product Manager organization identify the work that is viable, feasible, and sustainable to meet the customer's needs? Here are some ways:

- Market Research: Identify customer needs and market trends
- Product Strategy: What do you want to achieve, how do you plan to get there, and which products align with our core goals
- Roadmap Development: Outline the direction, priorities, and progress of a product over time
- Product Lifecycle Management: How to handle your product as it moves through development, introduction, growth, maturity, and decline.

For each of these, I want to share takeaways on how we could use these in our quality program to launch the most relevant services.

### 2.1.1 Market Research

A Senior Product Manager once told me his favorite interview question he liked to ask PM candidates. It was "*How do you design an elevator?*" The number one thing he wanted to see before anything was designed was for the candidate to ask, "*What is the building, and who is actually using the elevator?*" The idea is that you shouldn't be building products or services unless you understand whom you are building

them for. In the same sense, who and what is your quality program building for? As QA, we often assume that our customers are business customers. But have we considered our internal customers, the engineers? Do we understand their needs? In my example above about Microsoft, “*Full testing would take the better part of a day to run, many more hours to “analyze the results” to identify false failures, and days or weeks to repair all the tests that were broken due to some legitimate change in the product. (...) Two years ago, we started on a path to redo testing completely.* [1] “ The Microsoft QA team’s product strategy was to churn out as much end-to-end test coverage as possible, but in the end, this wasn’t what the engineers needed, and eventually, all that testing was replaced. Before investing energy in a strategy, have we interviewed our developers to understand what they are looking for in a quality partner?

One example I can give of understanding our customer use case before building the solution was a case back at a smaller startup I worked for. Our Feature Flags management was out of sync from staging to production at this company. This affected quality since we were testing a different product in our testing environment versus (flag on) what was out in production (flag off) due to the disparity of feature flag states. Leadership wanted to invest in a third-party tool to manage this or try to build some tooling to manage the flag states, but both solutions would take quarters to implement or were costly. Meanwhile, our builds became increasingly flakey due to needing to know what state of the build we were validating against. Talking to our developers, most of them mentioned that it would be simpler to track states through Jira and have QA modify the testing config files to reflect what was being changed. This was a bit of added work to the QA team, but taking on the ownership of Feature Flag states and maintaining our environments instantly allowed us to reduce flakey tests. By researching what our Developers really wanted, our QA program saved our company money, improved engineering bandwidth, and allowed QA to take on some work that had an immediate impact, even though it wasn’t a traditional project a QA program would normally do.

### **2.1.2 Product Strategy**

As a Quality Organization that provides services, having a product strategy ensures that the services we provide will actually achieve the goal the organization set out to achieve. For each product we build (for example, testing framework, new process, testing artifact, etc.), have we detailed our outcome and why the work done will achieve that outcome? Product strategy entails diving deeply into these two questions and ensuring what we create achieves our goals.

For example, if our service is end-to-end testing, I often hear that the goal is to provide high test coverage. However, at Iterable, the end business goal is to have very little disruption to our customer’s experience and not high test coverage. I was asked once by a new engineering manager what our end-to-end testing feature coverage was. He was beyond shocked when I mentioned it hovered at about 30%. However, looking at telemetry for the high amount of traffic Iterable receives over our features, we have very few customer-reported bugs. We can maintain this by doing two things well. We utilize Heap.io to give us data on the most trafficked portion of our application so we know that the most critical areas have strong end-to-end coverage. The second thing is release management, and part of our Quality Engineers’ job is to create a checklist that, before we release, we have monitorable dashboards and a gradual rollout strategy (Feature Flags or Cluster Deployments). If a defect is released, we can quickly detect and roll back that defect. In my example, our quality programs’ product strategy aimed to ensure customers had a good experience (low bug report rate) and did not have high test coverage.

### **2.1.3 Roadmap Development**

One of the most critical behaviors of a successful product manager is to be able to determine how complete a product needs to be before it’s ready to ship. The capacity to manage scope while meeting

deadlines is critical for success. Sometimes, the time to market will trump feature richness. In the same lens, if we build tooling, sometimes a quick proof of concept to bake with developers is good enough. I once had an engineer tasked with migrating us from a legacy testing framework to a newer one. His estimation would be two-quarters of the work, but I challenged him to partner with a scrum team that would use this tool and build out only three supercritical use cases that could be delivered in half a quarter. The result wasn't fully production-ready (it needed to be manually kicked off to run, there were not a lot of tests, there were no dashboards, etc.) but could be immediately run to gain some validation for critical use cases. Because it was adopted so quickly, we got a lot of feedback that ultimately changed our requirements for this tool. Today, we've gone back and added a lot of functionality, including full integration to our CI/CD pipeline, multiple integration libraries that support other scrum teams, and more test cases. These features were used by 80% of our company and have a positive sentiment with our developers. By ensuring we adopted the pilot and generated feedback quicker, we built a tool that was more used across engineering.

#### **2.1.4 Product Lifecycle Management**

One of the most essential directions a product manager needs to consider is the lifecycle of their product or services. They have to manage the product launch and ensure its adoption and growth. While the product matures, what tweaks are necessary to keep it relevant? Finally, when it is time to sunset the product, how can it be done gracefully? Any process or tooling we build should be treated in the same way. When we launch something, do we have a strategy on how we want the rest of the company to adopt it? Are we measuring the growth in usage? Did we account for the effort we need to drive usage across engineering? At Iterable, we launched a new quality initiative requiring engineers to fill out a release checklist to ensure a successful launch. However, we didn't create an easy way to track compliance and governance. Because of this, many teams just ignored the checklist and deployed in noncompliance. In the end, we didn't achieve our goal since we didn't have a strategy for adoption.

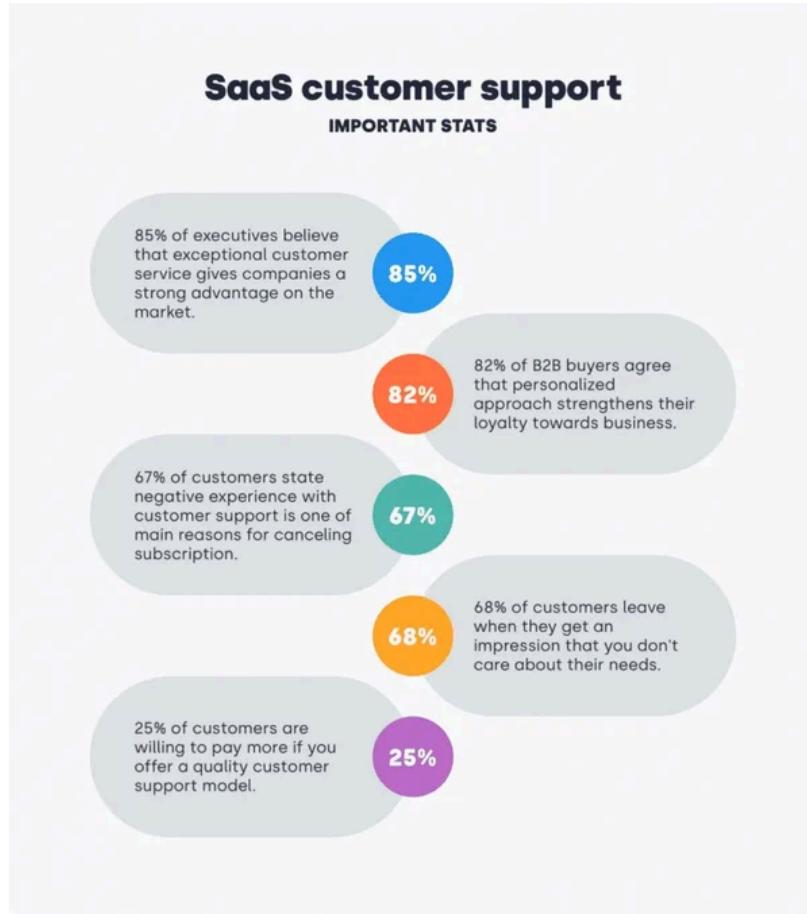
Another example was when we had a legacy testing framework running in CircleCI, which cost us extra credits. At the end of the day, we realized the framework wasn't giving us that many benefits for the costs, and we retired it. If we had strategized our sunset or when we should retire a framework, we would have saved the company some money and sped up our deployment time. Having a clear lifecycle management of all the Quality products you enact is critical for its long-term success.

### **3. The Problem: Engineers Don't Trust Our Work**

A few years back, I read a great article by Matt Eland called Partnering with Quality Assurance. This quote always makes me laugh because of how accurate it is: "*I've worked with some very devoted quality assurance (QA) professionals. I've also worked with some extremely talented developers. But when you put the two of them together – look out! When QA and development interact, sparks tend to fly, arguments start suddenly and swiftly, and things devolve into chaos.[2]*" The work our QA Program produces and the services we provide hold little value if our internal customers and the engineering team have low trust or use for them.

In the same way, how do SaaS companies deal with customers who do not trust their product or are struggling to utilize it correctly? If you are using a SaaS company product, and it isn't working as expected, you will create a ticket or reach out to your customer service manager from their customer service organization. This Customer Service organization aims to help its customers utilize the company's products and services. In fact, customer service is a critical function of an organization. One study found

that 7 out of 10 customers stopped doing business with companies because of poor support. Look at the following stats from a study on Slideshare [3]:



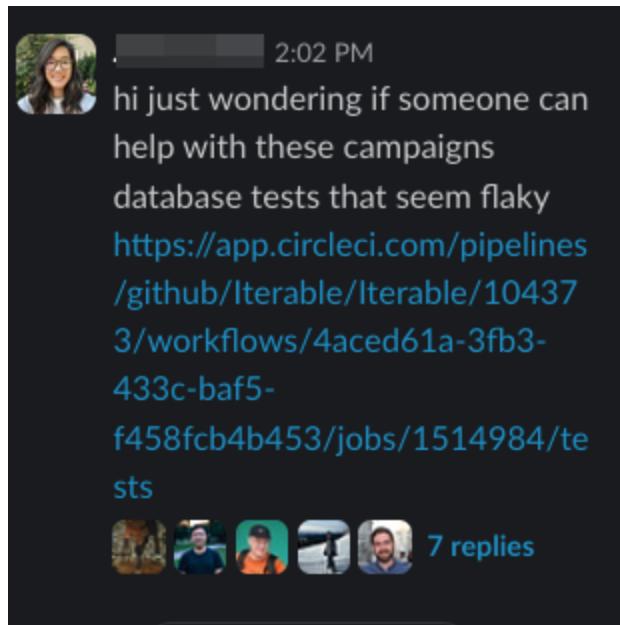
Looking at these stats, it becomes clear that we need to do more outside of testing to ensure our engineering customers are getting their perceived value. Managing this expectation and support will help guarantee that the engineering organization as a whole will have a higher value proposition of what the quality organization is providing to the engineering organization as a whole.

### 3.1 What are some examples of customer support in our quality program?

Here is how I have employed a few different services to provide support to our engineering team and leadership:

#### 3.1.1 Quality Rotation

Each quality engineer at Iterable is in an on-call queue and goes on-call for a week at a time. During this on-call rotation, they are tasked with triaging flakey tests, triaging broken builds, responding to incidents to help engineers, and attending incident retrospectives. Even though this can be time-consuming, it gives engineers confidence that the Quality team understands their pain points with the testing infrastructure and gives quality a voice during incident management. It also ensures that when something isn't right, our engineers feel like they can reach someone in QA quickly to start a resolution.



### 3.1.2 Open Ticketing System

We utilize an embedded QE model at Iterable, which means that only some teams have a dedicated QE. However, we do not want engineers without a dedicated QE to feel like they aren't supported, so we utilize Jira to allow engineers to request testing support. We have dashboards that show open tickets and how long they have been open and assign QE to help. When we first launched this system, we spent a lot of time advertising it, and now we see tickets created regularly, so we definitely know we have strong adoption.

### 3.1.3 QA Office Hours

Another service we provide to support our engineers is open QA office hours for engineers to discuss anything testing-related. We get questions about test plan creation, pairing on writing automation tests, and building infrastructure discussions. This open forum is nice to have, and some of the discussions can bubble up to leadership to drive higher-level engineering initiatives.

## 4. The Problem: The Quality Engineering Program Isn't Perceived as Essential

As a leader of a Quality Program, we must justify the cost of retaining our Quality Engineers and showcase the value they bring to the company. This challenging task is summed up quite well by David Caudill in his article *Maybe Getting Rid of Your QA Team Was Bad, Actually*. Mr Caudill writes:

*"The end result of this was that the slowest part of software delivery is testing. Since testing is why continuous delivery exists, that should have been good enough. Yes, we can make our tests faster, more automated, parallelized, etc. But when the highest value activity of a given practice is the bottleneck, you're optimal. You have achieved "the best possible problem".*

*Those habits and behaviors of optimization didn't stop there. We kept on chopping. We squished our integration and end-to-end tests down to unit tests to parallelize. At the personnel level, we pushed out*

*anybody whom we believed could not code, indiscriminately, at the function level. We decided that testing might not be the bottleneck...the QA team was. The industry began to treat the people in these roles worse and worse. Expectations for them went up, salaries went down(while everyone else's seemed to be going up!), we contracted the role, we offshored it, pretty much anything we could do to try and stop employing QA Engineers. [4]*

The problem here is when Quality Programs tie their value proposition to testing, and if the engineering organization doesn't see the value, or worse, sees the Quality Program as a detracting value, it becomes a massive target for the chopping block. But in reality, the Quality Program does much more than testing, and its value goes beyond the testing pipeline. How do we show this value proposition? All SaaS companies have an entire organization devoted to this very task, the Account Management Team. Let's see how they accomplish this.

## 4.1 Account Management for Quality Services

One of the most important organizations in any company is the account management team, which ensures that the customer is getting serviced well to increase their consumption of our product/service, maximize retention, and upsell more opportunities. By helping the customer understand the value they are getting from utilizing your service, account management ensures that the customer continues to use our services and opens up opportunities to use new services. Gartner [5] found that 82% of customers will renew when presented with a chance to switch if the company engages with them and performs value enhancement activities, which makes them feel like they used the product better and are more confident in their purchase decision.



As a Quality Program, we want to do something very similar. We want our engineering leadership team and engineers to understand the value they are getting from our Quality Program. If they understand the stories and see the telemetry of what is going well, they will be more likely to use our tooling or process and invest more in the Quality Program. Here are some ways that I've used to show the value of the Quality Program:

### 4.1.1 Monthly QA Reports

Every month, our Quality organization publishes a monthly QA report that contains our quality KPIs (Number of escaped bugs, Change Failure Rate, Number of incidents, Build Pipeline health, etc), success

stories that highlight what scrum teams are doing well, or failures and deep-dive root cause analysis over what happened. The goal of the report is twofold:

- Highlight Wins - Example: Our end-to-end testing framework caught four major defects, which prevented incidents
- Direction for Losses - Example: We had a new incident due to a change in the reporting page. We discovered that our test coverage was lower there, so we are adding more testing and monitoring so that this won't recur.

The goal is for each Quality Engineer to present the report's highlights to their scrum teams and for me to deliver this during our weekly engineering leadership meeting so the organization as a whole understands the value the Quality Program is delivering.

#### **4.1.2 Demo Days**

Every quarter, our engineering organization has a demo day. The QA organization always sends a member to demo something that the team is actively working on. When we demo, we utilize this template:

1. What is the tangible outcome of this project in relation to our engineers or customers
2. What impact will our customers feel once this is complete
3. What was the effect if we didn't do this
4. How do we measure the effectiveness once this launches
5. The technical portions underlying what we built.

The goal of presenting our demos using this method is to clearly communicate the value of what we build to our customers and make it clear that we are tracking how much our features are being adopted.

#### **4.1.3 Quality Champions**

One of my engineers developed this concept, and I love it. Once a month, we identify engineers who are doing excellent quality work (ex, found the most bugs during a bug bash), engineers who are power users of our quality services (ex, first engineer to utilize our library to mock customer data in a test), or engineers doing good quality work (ex: engineer that closed the most bugs and updated our dashboard). We then write up a blurb about what they did and why it's so impactful. Then, we post it on our company's kudos page. We also send the commendation to their managers and send them a custom Quality Champion T-Shirt. We find that if we incentivize and celebrate engineers who buy into our quality system, we will start to build champions for our program, which means more engineering interaction and value are created.

## **5. The Problem: Engineering Leadership is Not Investing in Quality Engineering Growth**

Do your Quality Engineering programs sell your quality services to senior leadership (technical tooling, process, culture shifts, etc.)? Do they lay out a strategy on why growth in the Quality Engineering program is necessary to expand these services? Quality Programs generally have no problem following their charter and executing well, but are they utilizing the right muscles to justify growth? Does your Head of Engineering understand why they must continually invest in the Quality Program?

John Warrillow, author of *The Automatic Customer: Creating a Subscription Business in Any Industry*, writes, "*Your biggest competitor for your subscription business is not the rival service; it is your customer's*

*inertia in not using your service. [6]*" SaaS companies also struggle with getting their current customers to understand why they should not just continually use their products but upgrade and expand in utilizing more services (and generate more profit!) Like SaaS companies, our quality programs also need some sort of sales motion to advocate for themselves continually.

## 5.1 Selling Quality Engineering as a Service

So, how do we ensure our leadership team is bought in on our growth strategy? Here are two ways I utilize my sales pitch: to explain why they should be bought into our quality services and why these services need to grow.

### 5.1.2 Quality Engineering Service Delivery Review (SDR)

Once a quarter, we have a high-level quality service delivery review with our engineering leadership team and product and customer support leadership. During this review, I reviewed the data similarly to our QA reports but overlayed the current in-flight projects and their impact on this data. This is how I show the value of what we are doing. Then, I showcase areas where we are doing poorly, the effect if we don't alleviate these, and what initiatives the QA Program can execute to make an impact. I present our upcoming roadmap, the deliverables we will be doing, and what the outcomes will be. Finally, I always treat the last portion as a sales pitch to the leadership organization. I tier the results, showing what we can accomplish with the current resources, what we can achieve if we add resources and scale, and the impact of each decision.

### 5.1.3 Engagement with Leadership Organization

As the leader of my Quality Program, I utilize repetition to help sell its value. I aim to meet with each member of the senior engineering leadership to understand what areas they are concerned about, present them with data (QA Report, QA SDR), and discuss how the Quality Organization can help them. This relationship is essential to cultivate to ensure I align with the quality program, which is working and giving value to my leaders. This also allows me to understand how each leader plans to scale their team and will enable me to pitch them the utilization of more of our Quality Services. Doing this helps secure internal champions for the Quality Program and its growth.

## 6. Conclusion

The role of Quality Programs must evolve outside of just driving quality to reframe itself as a broader, value-driven service. By mirroring some of the operational strategies of SaaS companies, our Quality Programs will be successful in highlighting how indispensable they are in delivering business value. By understanding our internal customer needs, robust quality product management, proactive customer support, and effectively communicating success and areas for improvement to senior leadership, Quality Programs can secure their relevance within engineering organizations today.

## References

1. Harry, B. (2017, June 28th) How we approach testing VSTS to enable continuous delivery. Retrieved from: <https://devblogs.microsoft.com/bharry/testing-in-a-cloud-delivery-cadence/>
2. Eland, M. (2019, Nov 29th) Partnering with Quality Assurance. Retrieved from: <https://killalldefects.com/2019/11/29/partnering-with-quality-assurance/>

3. Raileanu, G. (2016, Aug 18th) The Leading Reasons for Customer Churn in SaaS. Retrieved from:  
<https://www.slideshare.net/slideshow/the-leading-reasons-for-customer-churn-in-saas/65129018>
4. Caudill, D. (2023, Dec 3rd) Maybe Getting Rid of Your QA Team was Bad, Actually. Retrieved from:  
<https://davidkcaudill.medium.com/maybe-getting-rid-of-your-qa-team-was-bad-actually-52c408bd048b>
5. Gartner. (n.d.) How Service Leaders Can Increase Customer Loyalty. Retrieved from:  
<https://www.gartner.com/en/customer-service-support/insights/customer-loyalty>
6. Warrilow, John. The Automatic Customer: Creating a Subscription Business in Any Industry, 2015.

# Software and Methods Supporting Legal Proceedings: Home-Brewed and Speculative or Verifiably Reliable and Definitive?

**Kalman C. Toth** Ph.D. P.Eng.

[kalmanctoth@gmail.com](mailto:kalmanctoth@gmail.com)

## Abstract

Software-based tools are often used to collect evidence for court proceedings. Some are well-tested and commercially available. Others are not. Over the last fifteen years, tens of thousands of lawsuits claiming copyright infringement using BitTorrent networks have been brought by media companies. The software-based tools used to report copyright infringement are not commercially available, not qualified by third parties, yet promoted as “forensic tools”. Explained is how wrongfully accused users are harmed; how Daubert [6] addresses admissibility of evidence; how BitTorrent seeders share infringing movies in pieces with peers; how such BitTorrent monitoring software collects pieces of movies from IP addresses; and why IP addresses do not distinguish infringers from non-infringers. The tools collect only a few pieces of movies from targeted IP addresses, while no pieces are collected from seeders holding infringing copies. The methods speculate about infringement rather than definitively collecting complete playable copies for validation. The software does not overcome critical failure modes including abandonment, piece-unavailability, space-depletion, and choking. Explained is how repeatability and reproducibility testing, completeness, playability, transaction validation, and audio-visual matching yield verifiably reliable software and infringement reporting for acceptance in software engineering and forensics communities. Using unreliable software for forensics purposes is malpractice.

## Biography

*Kal Toth provides technical expertise to law offices defending individuals claimed to have used BitTorrent to infringe copyrighted movies. He has published several papers and US patents in the field of digital identity. Kal has worked for Hughes Aircraft, Datalink Systems Corp., the CGI Group Inc., the Software Productivity Centre (BC), Intellitech Canada (Ottawa), and the Canadian federal departments of Defence, Communications, Transportation, Revenue, External Affairs, and the Communications Security Establishment. He is former Executive Director of the Oregon Master of Software Engineering (OMSE) program, obtained his Ph.D. in computer systems and electrical engineering from Carleton University, and is a registered professional engineer with a software engineering designation in British Columbia.*

## 1. Introduction

BitTorrent [1] is used to legally share software, movies, games, and other content over the Internet - BitTorrent is also used to illegally share copyrighted movies. Defective methods and software monitoring BitTorrent for copyright infringement inflicts much harm on wrongfully accused users exposing them to significant litigation cost, stress, lost privacy, and damaged relationships. Prenda Law<sup>1</sup> used supposedly reliable BitTorrent monitoring software to claim thousands of users had infringed. Likely many users were harmed. Evidence of self-seeding collapsed the case and bankrupted the firm.

NARS [14], MaverickMonitor [15][16] and VXN Scan<sup>2</sup> [17] are tools routinely used to monitor BitTorrent usage and assert IP addresses were used to infringe copyrighted movies. Owners refer to their tools and methods as *infringement detection systems* and/or *forensics tools* and claim the implementing software is free of defects (reliable) and reports infringement accurately. Owners provide no evidence supporting such claims and normally decline to disclose the source code. In one case, however, the source code was provided and author comments suggested numerous bugs remain latent in the software.

This paper discusses challenges BitTorrent monitoring tools do not overcome, and explains how software engineering and forensics standards can be exploited to overcome most (but not all) of them.

## 2. Copyright Infringement in BitTorrent Context

According to the US Copyright Office, copyright infringement occurs when a copyrighted work is reproduced (i.e. copied), distributed, performed, publicly displayed, or made into a derivative work without the permission of the copyright owner [2]. With respect to movies (e.g. MP4s, MOV, AVI, WebM), this implies that possession and playability by alleged infringers should also be demonstrated. After all, holding a file of unplayable zeros and ones does not infringe copyright. In the BitTorrent context, digital media is infringed when a person:

- holds a complete or substantial playable portion of a copyrighted movie obtained from BitTorrent, or
- distributes a complete or substantial playable portion of a copyrighted movie into BitTorrent.

## 3. Admissibility of Technical Evidence - Daubert

In the United States, the admissibility of computer evidence is typically introduced at a very late stage of the proceedings where both sides have incurred tremendous economic and emotional capital. Daubert [6, 10,12] addresses the admissibility of technical evidence in court proceedings. Daubert principles rely on the sufficiency of facts and data, reviews by peers, acceptance by the relevant scientific community, and qualifications of relied-upon experts. Daubert also implies software-based tools used to collect forensics evidence for court proceedings should undergo empirical testing and be presented and/or published at/in conferences, workshops, and journals/periodicals for review and acceptance by the relevant scientific community, in this case, the software engineering and forensics communities.

## 4. BitTorrent Clients Installed on PCs Share Movies in Pieces

To share content over BitTorrent (BT), users install software applications called BitTorrent clients on their personal devices. These clients implement the BitTorrent protocol specification [1] enabling peer-to-peer (P2P) sharing by way of IP addresses. BitComet, BitTorrent, Deluge, qTorrent, rTorrent, uTorrent, and Vuze are popular BitTorrent clients. Currently, uTorrent<sup>3</sup> has the largest market share.

---

<sup>1</sup> [https://en.wikipedia.org/wiki/Prenda\\_Law](https://en.wikipedia.org/wiki/Prenda_Law)

<sup>2</sup> Excipio operates NARS (Network Activity & Recording Supervision); MaverickEye owns MaverickMonitor; Strike 3 owns VXN Scan

<sup>3</sup> uTorrent has about 69% of the market for BitTorrent clients

BitTorrent clients enable their owners to share movies and other media in small “pieces” using their IP addresses. A BitTorrent swarm is composed of IP addresses sharing pieces of the same movie. A BT client holding all pieces of a movie (a complete playable copy) is called a “seeder” or “seed” while a BT client interested in obtaining pieces of the movie is called a “peer”. Swarms sharing popular movies can attract 100s, 1000s or more peers. “Have”, “interested”, and other such protocol messages are used by seeders and peers to share pieces. Peers can obtain pieces from seeders (seeds) or from other peers. Peers refusing to contribute pieces to a swarm, called “leechers”, can be “choked” from obtaining pieces thereby throttling the rate at which pieces are obtained. The hash of each piece is verified when received. Once a peer has obtained and verified all pieces of a movie, the BitTorrent client assembles the pieces into an exact playable copy. Peers obtaining complete playable copies can choose to become seeders actively uploading pieces to peers in the swarm, or they can leave the swarm (become inactive).

Figure 1 depicts ❶ a 47-minute 391 MB playable movie (#m) fragmented into 746 x 524,488B (1/2 GB) pieces by the initial seeder’s BT client. Also depicted is a user with a BT client ❷ sharing a router and IP address wishing to obtain a copy of movie #m. The seeder’s BT client creates a .torrent file (.torrent m#) specifying certain characterizing attributes (metadata) of movie #m; a pointer to the file holding the movie; and a SHA1 hash (160-bit/20-byte) of all 746 pieces (piece-hashes) of movie #m. The .torrent file itself is uniquely identified by a SHA1 hash, infohash #m, calculated from the .torrent metadata and the 746 piece-hashes. Infohash #m identifies the BitTorrent swarm sharing the pieces comprising movie #m.

The initial seeder uses her client to post ❷ .torrent #m to Torrent Site(s). When ready to share pieces of movie #m, the initial seeder sends ❸ infohash #m and her IP address to a BitTorrent Tracker which uses infohash #m to track IP addresses participating in swarm #m. Peers interested in movie #m search ❹ Torrent Site(s) to locate and obtain a copy of .torrent #m. To obtain pieces from swarm #m, peers use .torrent #m to calculate infohash #m, send infohash #m and their IP addresses to the BitTorrent Tracker. The BitTorrent tracker returns IP addresses of seeders and peers sharing pieces in the swarm.

User ❺ progressively receives requested pieces ❻ from seeders and peers over the movie timeline. The user’s BT client hashes received pieces verifying they match the .torrent #m file’s piece-hashes. Once all pieces are received and verified, they are assembled into a playable copy of the shared movie. Peers with complete playable movies can opt to seed ❼ pieces they hold back into the swarm ( $p^1 \Leftrightarrow s^1$   $p^2 \Leftrightarrow s^2$ ).

## 5. Monitoring Software Collects Pieces from IP Addresses

A software-based BitTorrent monitoring tools exploit Torrent web sites, .torrent files, BitTorrent trackers and the BitTorrent Protocol [1] to locate swarms sharing pieces of copyrighted movies of interest. Infringement reports produced by such tools are similarly formatted. BitTorrent monitoring tools mimic peers downloading pieces from IP addresses but do not upload pieces to requesting peers (called leeching). Monitoring tools locate the .torrent file of a targeted movie and use the infohash of the .torrent file to obtain IP addresses participating in the swarm from a tracker. Monitoring tools record BitTorrent transactions in protocol capture files (PCAPs) including date/time, monitor’s IP address, client’s IP address, client ID, .torrent file infohash identifying the swarm, and the pieces transferred (in blocks).

Figure 1 depicts a BitTorrent monitoring tool comprised of a proprietary BitTorrent Client and a repository for storing PCAP files. Figure 1 illustrates the BitTorrent monitor ❽ obtaining the .torrent #m for movie #m from a Torrent Site; ❾ calculating infohash #m from .torrent #m; and using infohash #m and the monitor’s IP address to retrieve IP addresses of users participating in swarm #m from a Tracker; and ❿ collecting pieces of movie #m from selected IP addresses in swarm #m.

There is no evidence owners have demonstrated or otherwise established their BitTorrent monitoring tools conform with, meet, or satisfy guidance, recommendations and/or standards of the software engineering community (e.g. SEI/IEEE [3] [4] [5]) or the forensics community (e.g. NIST [6] [7] [8]).

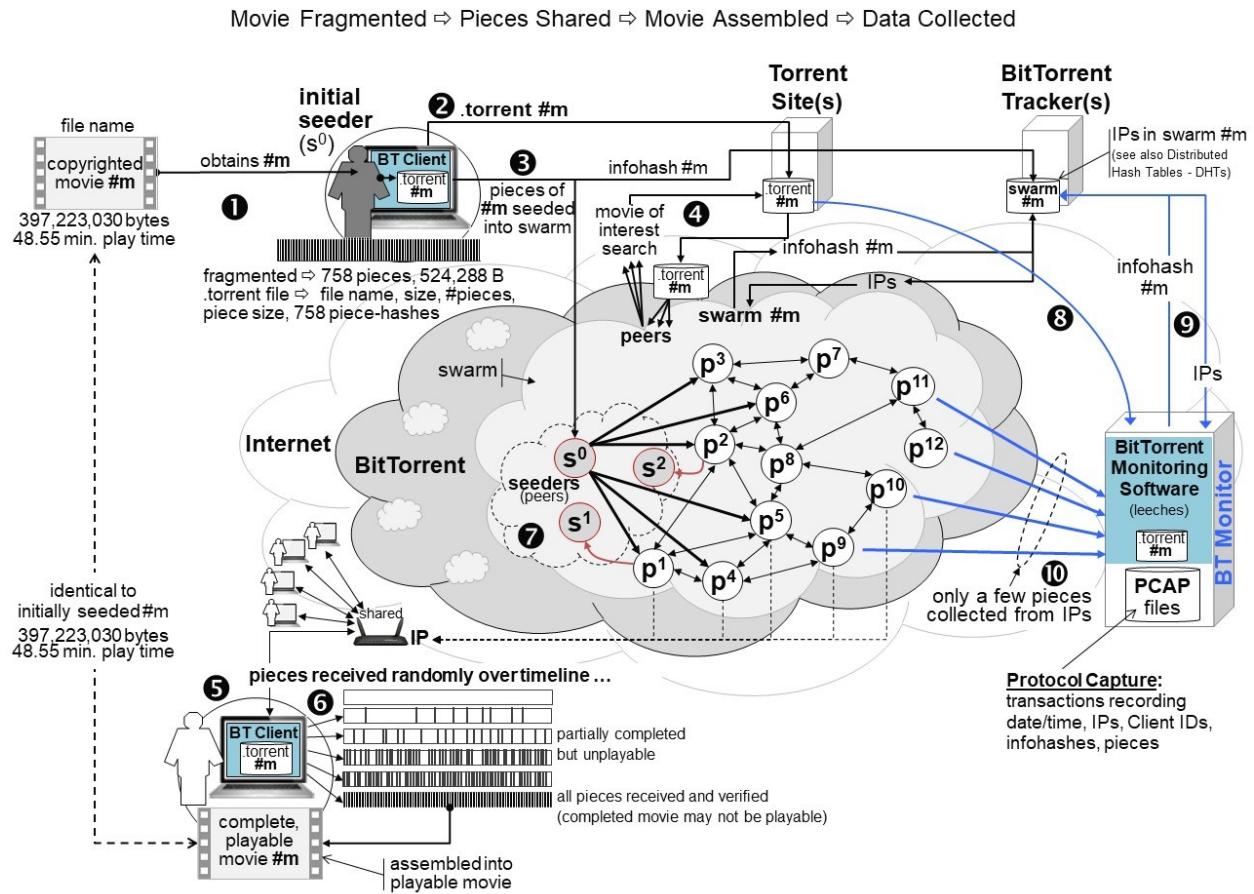


Figure 1. BitTorrent Sharing, Monitoring, and Protocol Capture

## 6. Swarm and IP Address Targeting Not Validated

Figure 1 ❸ does not explain how BitTorrent monitoring tools target .torrent files and swarms suspected of being used to actively share copyrighted movies.

Apparently, .torrent files thought to be used to share pieces of a *copyrighted movie* are searched and then audio-visually matched. Since movie titles, actor names, and attributes may be misspelled or modified, search terms may need to be tweaked until a prospective swarm is identified. Pieces are collected from multiple IP addresses participating in a swarm until a playable *full copy* is obtained which is audio-visually matched to a *copyrighted movie*. If successfully matched, the swarm is asserted to be sharing infringing pieces of the *copyrighted movie*. Subsequently, the swarm and IP addresses participating in the swarm are targeted by the BitTorrent monitoring software.

*However, PCAPs validating the swarms from which such full copies are obtained are not recorded. There is no evidence that targeted and monitored swarm share pieces of copyrighted movies as alleged.*

## 7. IP Addresses Do Not Identify Infringers

Shared routers are universally used to access the Internet over public ("routable") IP addresses allocated by Internet Service Providers (ISPs) to the router. There is no evidence BitTorrent monitoring tools can trace transactions to and from private IP addresses of shared routers to: (a) unambiguously identify

persons using private IP addresses of a router; or (b) determine which private IP addresses are being used by users with a BitTorrent client; or (c) determine whether only a single private IP address of the router is being used; or (d) if the subscriber is the only person using the IP address at a given date/time.

Consider a household of 2.5 family members<sup>4</sup> plus a guest and a neighbor sharing a router and IP address. Assume that of these 4.5 persons, one of them used the IP to infringe a copyrighted movie. It is only 22% likely a given user sharing the router infringed, while it is 78% likely an arbitrarily selected user, including the subscriber, did not infringe. Yet courts have routinely issued subpoenas to ISPs to disclose the identity of the subscriber of an IP address reportedly used to infringe a copyrighted movie.

**Figure 2 ④ illustrates a critical limitation:** Shared IP addresses do not identify the person or persons (including the subscriber) who may have used a shared IP address to infringe copyrighted movies over BitTorrent. Cobbler<sup>5</sup> [11] concluded that IP addresses alone do not identify infringers [12].

## 8. Speculative Methodology – Failure Modes Ignored

Tool owners routinely speculate that collecting a few pieces from an IP address in a swarm is enough to assert that an IP address was used to infringe a copyrighted movie. The BitTorrent monitoring methods and tools referenced in this paper typically collect a tiny fraction (2 or 3 pieces) of a movie from a targeted IP address which represents 0.3% or 9 seconds of play time<sup>6</sup> of an average-sized movie. As illustrated in Figure 2 ④ and ⑥, such minuscule evidence of infringement is not predictive of user actions and critical failure modes preventing the downloading and assembly of complete playable movies. Figure 2 ① ② ③ illustrates that BitTorrent monitoring tools discussed do not report the IP addresses of seeders who, by definition, hold all pieces of movies shared into a targeted BitTorrent swarm.

**Critical Failure Modes:** Critical failure modes include: BitTorrent users abandoning downloads before completion; unavailability of pieces in a swarm preventing completion; memory or disk space depletion causing the software of a BitTorrent client to fail; and BitTorrent protocol “choking” preventing BitTorrent monitoring software from collecting pieces held by a targeted IP address.

**Abandoned Downloads:** Users may abandon downloading pieces from a swarm before completion because of mistaken intent, lost interest, or frustration due to piece-unavailability. Since monitoring tools collect only a few pieces of movies from targeted IP addresses they do not detect abandonment.

**Piece-Unavailability:** Download completion and playability relies on seeders and peers making enough pieces available in a swarm and monitoring the swarm long enough to collect all the pieces held by a targeted peer and IP address. Piece-unavailability prevents completion and hence playability.

**Space Depletion:** Certain BitTorrent clients pre-allocate space for pieces to prevent failure due to space depletion; to eliminate the time required to render playable movies; and to increase download speed. Most BitTorrent clients do not support space pre-allocation thereby increasing the likelihood of software failure before completion especially when downloading high-resolution/feature-length movies.

**Choking:** Since BitTorrent monitoring tools emulate leeching peers they may be “choked” by peers for refusing to share pieces when requested. Choking can prevent BitTorrent monitoring software from collecting all pieces of movies from targeted IP addresses likely rendering them unplayable.

To prove such failure modes have not corrupted infringement reporting, BitTorrent monitoring tools must:

- Collect all pieces from IP addresses to prove downloads were not abandoned before completion.
- Increase the duration of swarm monitoring to achieve sufficient piece-availability, obtain all pieces of the movie from the swarm, and thereby avoid false infringement reporting.

---

<sup>4</sup> 2020 US census estimated that the average household has about 2.5 members

<sup>5</sup> Cobbler Nevada, LLC v. Gonzales, 901 F.3d 1142 (9th Cir. 2018)

<sup>6</sup> Based on movie size of 397 MB, piece size of 524,288 B and playtime of 48.55 seconds

- Collect pieces only from BitTorrent clients pre-allocating space (e.g. rTorrent and uTorrent) to reduce BitTorrent client space depletion risk and increase likelihood complete playable copies are obtained.
- Collect all pieces of movie from swarm to prove choked monitoring has not prevented completion.

*BitTorrent monitoring software must obtain a complete copy (all pieces) of a copyrighted movie to objectively prove that critical failures (abandonment, piece-unavailability, space depletion, and choking) did not occur when monitoring a given IP address in a targeted swarm.*

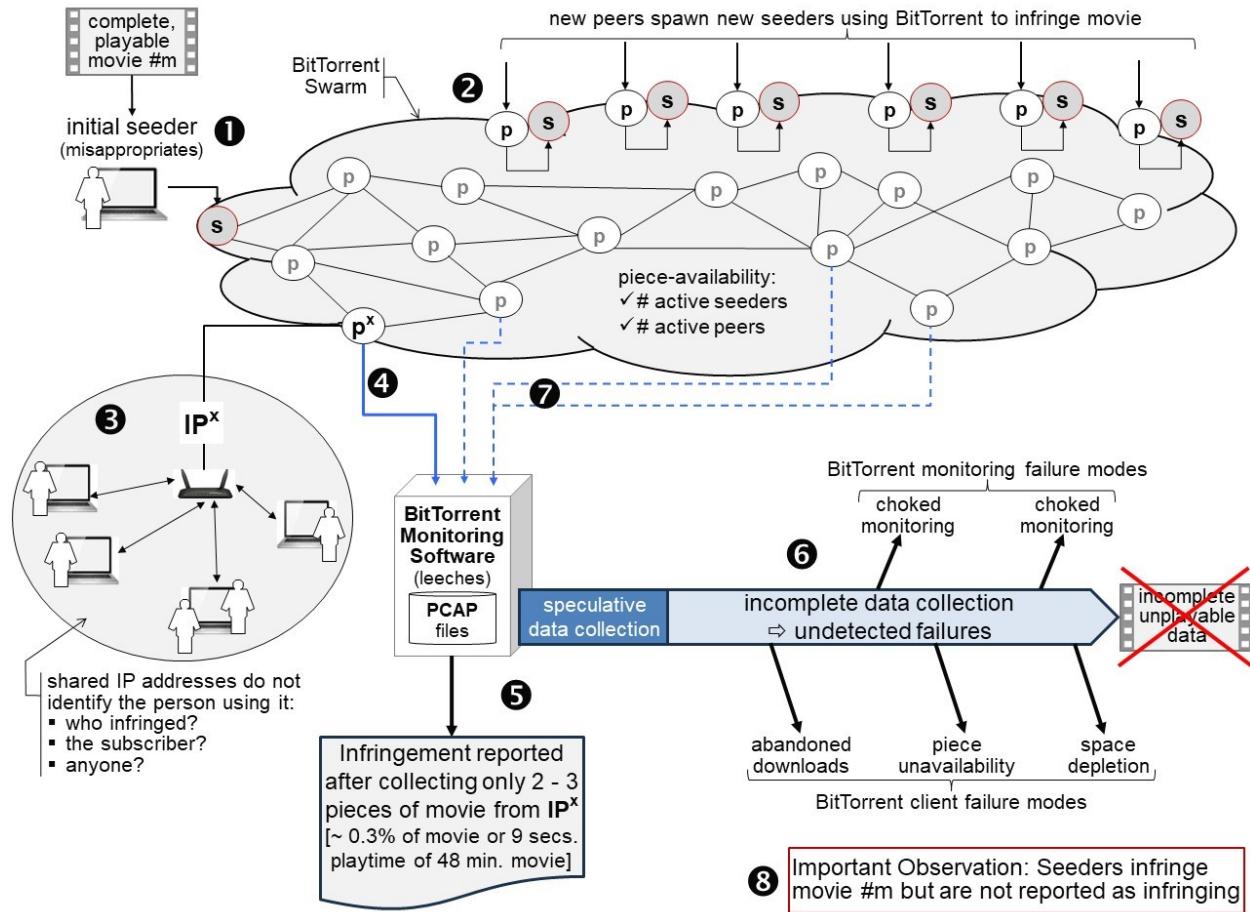


Figure 2. Illustrating Speculative Reporting, Failure Modes, and Unreported Seeders

## 9. No Evidence the Monitoring Software is Verifiably Reliable

Tool owners have claimed, without evidence, their monitoring software is free of defects. However, no technical documents are provided: no operational theory describing the forensics purpose; no requirements, architecture or design specifications; no use cases or UI specs; no user, operator, and maintenance manuals; and no records of peer reviews, code inspections or design walkthroughs. Tests have typically been conducted after infringement monitoring and reporting. If testing is not conducted prior to use, there is no evidence the monitoring software operates correctly when deployed.

*There is no evidence that the monitoring software is verifiably reliable before use.*

## 10. BitTorrent Monitoring Software Trivially Tested

Test procedures, test data and test results were undocumented. A few light-weight (trivial) informally described test cases were conducted using personal computers with installed BitTorrent clients and test movies ranging from a few minutes of play time to 45 minutes which are water-marked. Test computers (PCs) with installed BitTorrent clients were connected to the Internet by a fixed IP address. The BT client of each PC was used to create a .torrent file for each test movie. The tester then provided the infohash derived from each .torrent file to the system administrator enabling the BitTorrent monitoring software to download all pieces of each test movie. The test cases establish rudimentary swarms having (only) two participants - the tester seeding pieces into BitTorrent, and the monitoring tool collecting pieces.

The test cases were not conducted using shared routers and IP addresses allocated by an ISP. No tests were conducted to objectively verify that the monitoring software can identify persons using private IP addresses of routers and/or distinguish infringers from non-infringers using a given router. And the tests were not conducted under loaded conditions with 100s, 1000s, or more peers competing for pieces from seeders and other peers – in other words, there is no evidence BitTorrent monitoring software can download all pieces of a movie under such loaded conditions. Finally, no tests were conducted to verify the monitoring software can collect all pieces of high definition, feature-length movies from IP addresses.

*Apparently, comprehensive tests providing strong reliability assurances before use were not conducted. And there is no evidence that robust repeatability and reproducibility testing was conducted.*

## 11. PCAPs and Infringement Reporting Completeness

PCAPs should be exploited to validate the completeness of infringement reporting. The essential purpose of recording transactions (PCAPs) is to introduce redundancy to off-set defects that may be latent in the code, report defects and anomalies, and thereby elevate software reliability. Once BitTorrent monitoring software has collected all available pieces from a targeted IP address in a targeted swarm, the recorded PCAP files should be used to verify all collected pieces were obtained from the same IP address, and hashes of all pieces recorded match the piece-hashes specified in the .torrent file identifying the targeted swarm by way of the derived infohash.

*However, the monitoring software does not collect all pieces of a movie from IP addresses in a swarm, apparently the goal being is to identify the swarm's infohash and target all IP addresses in the swarm.*

## 12. Matching Infringing Copies to Copyrighted Movies

Ideally, upon successfully validating all pieces of a movie (complete copy) have been collected from a targeted IP address, the pieces would be assembled and rendered into a playable movie that would be audio-visually matched to the copyrighted movie apparently shared over the targeted swarm.

*Given only a few pieces are collected from an IP address, audio-visual matching is not possible.*

## 13. Inspected Code – Numerous Unresolved Bugs

In one case, source code was provided. Presumably it represented the operational software implementing the BitTorrent monitoring tool. No technical specifications or build/make files were provided. It was therefore not possible to establish if the provided source code compiled into the executable code. Code inspection revealed the software consisted of a collection of open-source software modules. No documents were provided explaining how the modules were compiled into the runtime, or how they were integration-tested. Author comments implied the software contained many unresolved bugs.

*No evidence the software was reliable, or the runtime code was compiled from the open-source software.*

BT Monitoring Software Assertions	Critical Issues
Swarm & IP Address Infringement Assertion: ▪ swarms located sharing infringed movies ▪ “full copies” obtained from swarms matched to copyrighted movies	PCAPs recording swarm from which a “full copy” obtained not recorded (full copies cannot be traced to swarms) ▪ hence no objective targeted swarms and IP addresses share the matched copyrighted movies as alleged
Accuracy Assertion: ▪ The software reports infringement accurately	IP addresses do not identify persons infringing movies: ▪ users share routers, PCs, and IP addresses ▪ monitoring software cannot distinguish infringers from non-infringers (including subscriber) sharing IP address ▪ add'l measures needed to identify infringers accurately
Piece Collection Sufficiency Assertion: ▪ collecting only a few pieces from an IP address is purportedly enough to assert infringement	Speculative forensics methodology ▪ 2 or 3 pieces (about 0.3%) of movies collected from IPs ▪ all pieces must be collected to avoid false reporting due to abandoned downloads, piece unavailability, BT client storage space depletion and choked BT monitoring
Reliability Assertion: ▪ The monitoring software is free of defects	No evidence monitoring software is verifiably reliable: ▪ no tech specs, manuals or test docs; no peer reviews ▪ no evidence of comprehensive / robust testing ▪ transaction records not used to validate completeness ▪ provided open-source contained many unresolved bugs
Testing Assertion: the monitoring SW “works” ▪ handful of test movies, test PCs, BT clients ▪ movies downloaded from a fixed IP address ▪ infohashes provided to administrator ▪ brief videos (min) and short films (45 min)	Trivial, light weight tests, not repeatable, not reproducible: ▪ no test procedures, no test data, no test results ▪ downloading from users sharing routers not tested ▪ not tested under loaded conditions [100s/1000s peers] ▪ not tested using high-def (HD) feature-length movies

Table 1: Assertions vs. Critical Issues

## 14. Analysis: Existing BitTorrent Monitoring Software

Existing BitTorrent monitoring software does not detect abandoned downloads, piece-unavailability, client space depletion, or choking. Such failure modes can be overcome by collecting all pieces of allegedly infringing content from targeted swarms and IP addresses. However, BitTorrent monitoring tools report IP addresses as infringing after collecting only a few pieces – such sparse data cannot be played (viewed). The methodology speculates about infringement rather than obtaining complete playable content that can be viewed and compared to allegedly infringed movies. Given the monitoring software is undocumented and was not tested under typical workloads using shared routers or feature-length movies, the software is not verifiably reliable. Finally, IP addresses alone do not identify infringers or distinguish them from non-infringers nor provide evidence that the subscriber had a BitTorrent client and used it to infringe a movie. Figure 3 presents a top-down analysis of BitTorrent monitoring software. The forensics process is split into three branches: the forensics methodology; software development; and implemented software.

- *IP addresses do not identify infringers or distinguish them from non-infringers or subscribers.*
- *The monitoring software is not verifiably reliable – no technical specs; no peer reviews; trivial testing.*
- *Daubert principles not satisfied – no peer reviews; no tech pubs; and no tech community acceptance.*
- *Speculative methodology – sparse data collection, no verifiable evidence of copyright infringement.*
- *Failure modes not overcome – abandonment, piece-unavailability, space depletion, and choking*
- *PCAP data and audio-visual matching not used to validate infringing IPs, completeness, playability.*

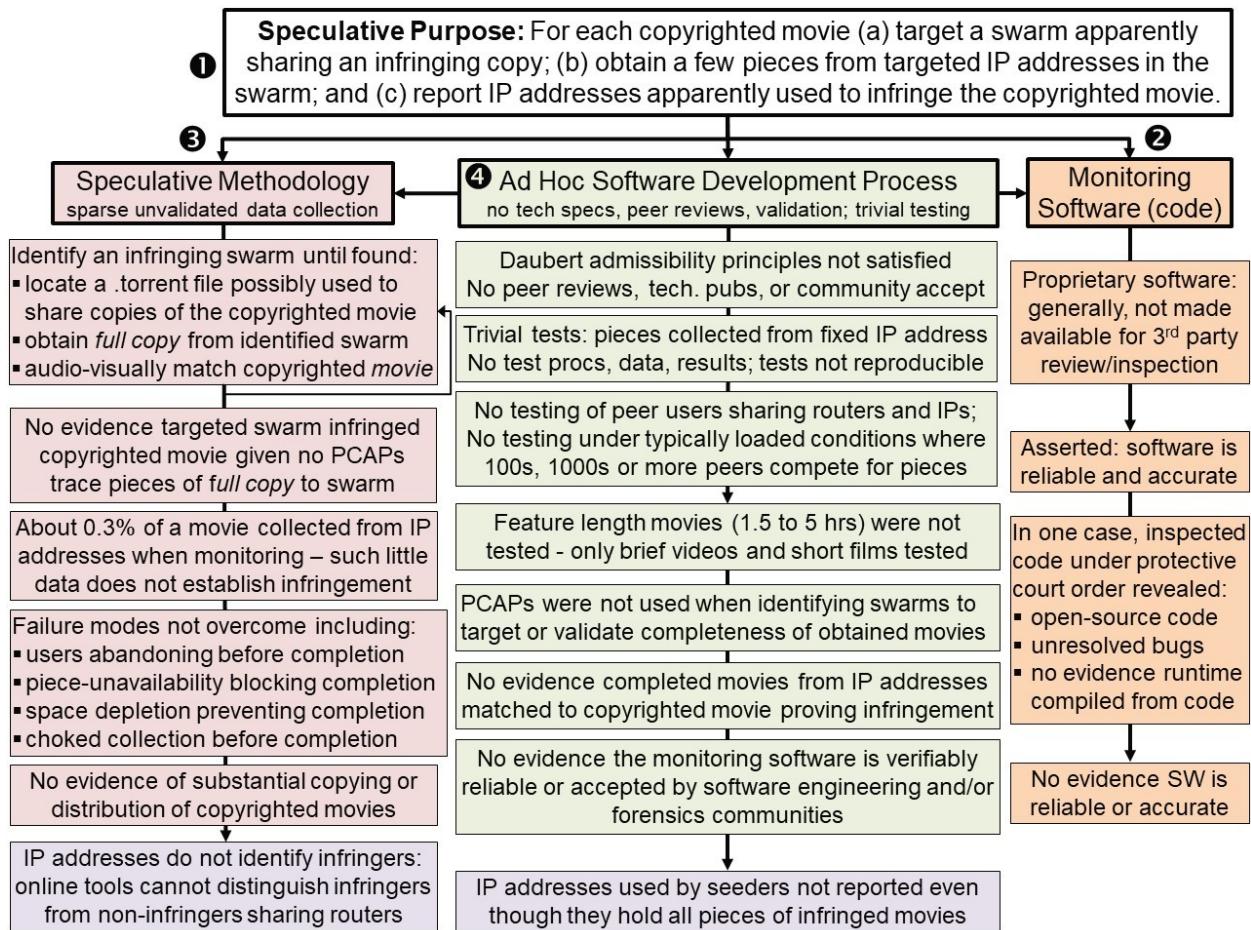


Figure 3. Analysis of BitTorrent Monitoring Software

## 15. Software Engineering Community Acceptance

Software engineering methods, guidelines and standards are prescribed by the Software Engineering Institute (SEI) [3] [4] and Institute of Electrical and Electronic Engineers (IEEE) [5]. There is no evidence referenced BitTorrent monitoring tools have been accepted in the software engineering community.

- Development is not guided by software engineering methods, standards and best practices.
- There is no theory specifying how infringement data is collected, inspected, analyzed, and reported.
- There are no technical specs of the design, critical functions, internal/external, or user interfaces.
- Apparently, technical specifications not reviewed by peers to verify software integrity and reliability.
- Trivial tests simply show monitoring tools obtain all pieces of test movies from a known IP address.
- Testing not conducted under typical workloads where 100s, 1000s, or more users share pieces.
- Tests were not configured with multiple peers using routers and sharing hi-def, feature-length movies.
- Critical failure modes are not tested (abandonment, piece-unavailability, space depletion, choking).
- There is no evidence critical functions were tested (i.e. piece-hashes, infohashes, movie rendering).
- Transaction records (PCAPs) are not used to validate IP addresses, completeness, and playability.
- No evidence infringing pieces collected from allegedly infringing IPs are matched to infringed movies.

*Acceptance by the software engineering community is unlikely without objective evidence attesting to the reliability and accuracy of BitTorrent monitoring software (software conferences, workshops and journals).*

## 16. Forensics Community Acceptance

As mentioned, BitTorrent monitoring tool owners routinely assert their tools are reliable and report infringement accurately. There is no evidence these tools have been presented or published at/in forensics conferences, workshops, or journals substantiating such reliability and accuracy assertions.

The National Institute of Standards and Technology (NIST) is a well-respected forensics authority:

- NIST promotes equity in the criminal justice system and strengthening of the scientific basis for forensics, namely, collecting, examining, analyzing, and reporting evidentiary data.
- NIST's Computer Forensics Tool Testing (CFTT) program establishes methods for testing forensics tool reliability including appropriate test procedures, test criteria, test sets, and test hardware [8].
- NIST maintains a searchable "Computer Forensics Tools and Techniques Catalogue". The BitTorrent monitoring tools referenced in this paper do not appear in NIST's catalogue.

NIST states [6] [7] forensics tools producing digital evidence admissible in court must produce repeatable and reproducible results. Repeatability implies the same test results are achieved in the same testing environment, while reproducibility implies the same test results are achieved in different testing environment(s). Test data and test results should be observable by 3<sup>rd</sup> parties.

In the current context, a .torrent file represents test data for a BitTorrent tool monitoring a swarm. Test results include reviewable transaction records (PCAPs) and playable movies.

- Repeatability tests are successful when all pieces of a movie shared across a swarm have been collected from an IP address, piece-hashes verified, and the pieces assembled into a playable movie.
- Reproducibility tests are successful when all pieces of a movie have been collected from different IPs in the same swarm, hashes verified, and pieces assembled into copies of the same playable movie.
- Repeatability and reproducibility testing should also be conducted across different swarms using different IPs (e.g. IP<sup>1</sup> & IP<sup>2</sup> obtain movie<sup>A</sup> from swarm<sup>A</sup>; and IP<sup>3</sup> & IP<sup>4</sup> obtain movie<sup>B</sup> from swarm<sup>B</sup>).
- The above tests should be conducted at different times under small, medium and large workloads.
- Test movies should range from brief videos to feature-length movies.

*There is no evidence existing BitTorrent monitoring software was repeatability and reproducibility tested.*

## 17. Software and Forensics Standards Elevate Reliability

Satisfying published software and forensics standards, guidelines and recommendations substantially increases the reliability of deployed BitTorrent monitoring software while also increasing the likelihood reported IP addresses were used to infringe copyrighted movies.

- Specify the definitive forensic purpose of the software implementing the BitTorrent monitoring tool.
  - Collect pieces from targeted IP addresses in a swarm until complete playable copies are obtained.
  - Employ a proven software development process to create verifiably reliable software.
  - Use shared routers and IP addresses, feature-length movies, and active swarms when testing.
  - Conduct repeatability/reproducibility testing across different IP addresses and different swarms.
  - Conduct tests over active (live) swarms with 100s to 1000s sharing short to feature-length movies.
  - Use transaction data (PCAPs) to validate test results and infringement data when monitoring.
  - Have qualified 3<sup>rd</sup> parties inspect PCAPs to validate IP addresses, completeness and playability.
  - Provide PCAPs and infringing copies of movies to defendants to enable them to validate evidence.

*Figure 4 identifies critical enhancements to the forensics methodology and BitTorrent monitoring software needed to achieve acceptance in the software engineering and forensics communities.*

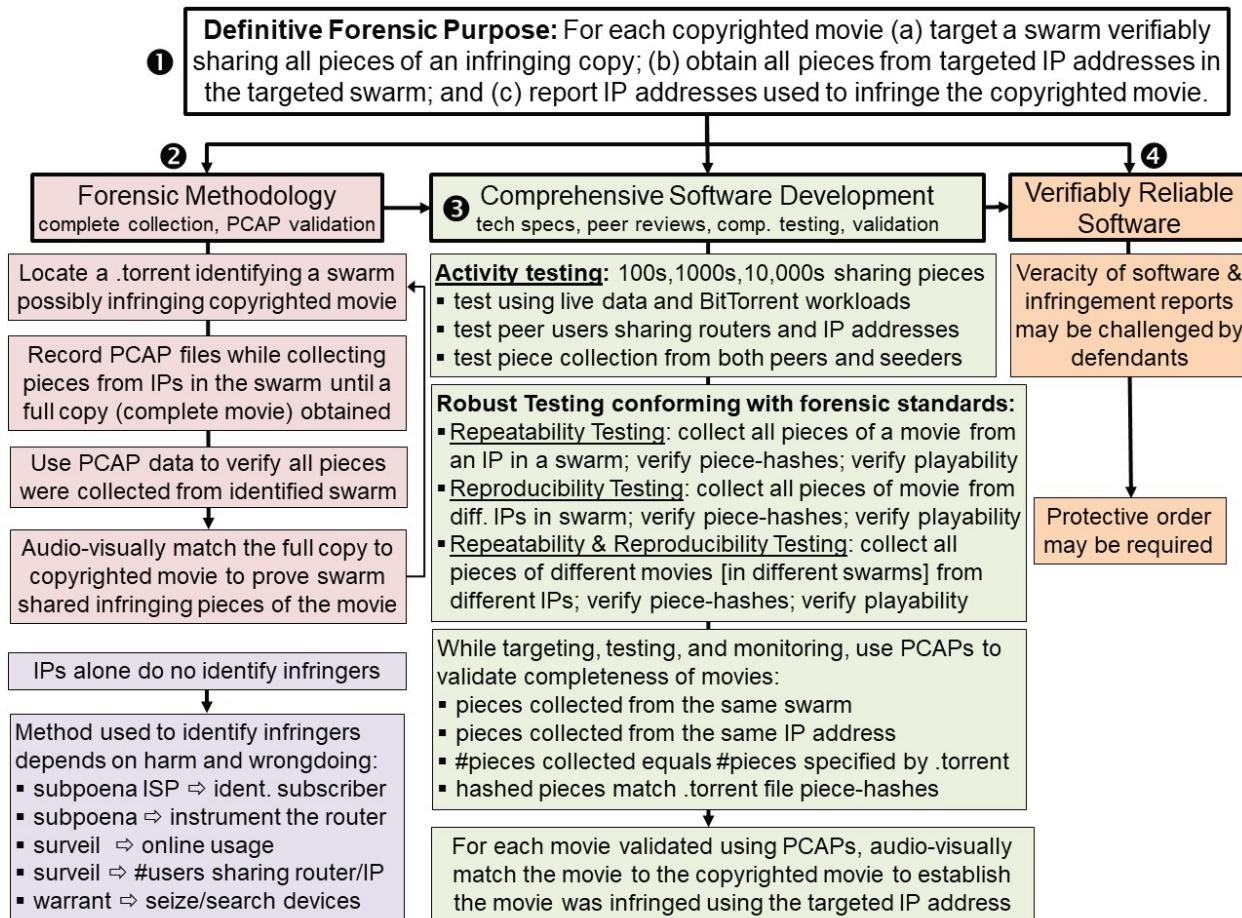


Figure 4. Software Engineering and Forensics Standards Elevate Software Reliability

## 18. Closing Remarks: Methodology and Software Reliability

**Home-Brewed and Speculative:** Without technical specs or evidence that proven development and testing processes and methods were used, one can only conclude the monitoring software is ad hoc “home-brewed” and the software is not verifiably reliable. Trivial testing without shared routers under lightly loaded conditions; and capturing transactions (PCAPs) without using them to validate infringement claims, significantly undercut reliability assertions. Collecting only 0.3% of a movie from an IP address participating in a targeted swarm does not represent substantial copying or distribution and speculates, without evidence, that the IP address holds the remaining 99.7%. Collecting such sparse data ignores the implications of abandoned downloads, piece-unavailability, space depletion, and choked piece collection each use case representing non-infringement.

**Verifiably Reliable and Definitive:** Targeted swarms are those thought to infringe a copyrighted movie. Repeatability and reproducibility testing across active swarms and IP addresses before use ensure the monitoring software is verifiably reliable. To overcome failure modes, monitoring software collects pieces from IP addresses in targeted swarms to completion. PCAP validation verifies that the hashes of all pieces collected from an IP address in a targeted swarm match the piece-hashes of the associated .torrent file. If PCAP validation is successful, audio-visual matching is conducted to verify the pieces collected from the IP address render a playable movie matching the copyrighted movie.

**Limitation:** *Verifiable evidence that a complete playable copy of a copyrighted movie was obtained from an IP address proves infringement, but it does not identify the person who used the IP address to infringe.*

## 19. References

- [1] *BitTorrent Protocol Specification v1.0*, <http://wiki.theory.org/BitTorrentSpecification>.
- [2] *U.S. Copyright Office - Definitions (FAQ)*, [www.copyright.gov/help/faq-definitions.html](http://www.copyright.gov/help/faq-definitions.html).
- [3] *Software Engineering Institute's (SEI) Capability Maturity Model (CMM)*: Assists organizations choose and tailor the most appropriate software behaviors, practices, and processes to achieve software reliably and sustainably goals.
- [4] *Software Reliability Tutorial*, 2011-2015 by Gullo and Peterson: Pages 25 and 29 tabulate empirically derived software fault rates across the five (5) maturity levels of the CMM determined by leading software reliability experts in the field (Keene, Jones and Krasner).
- [5] *IEEE Software Engineering Standards including IEEE Std 12207, Systems and Software Engineering Software Life Cycle Processes*: Documents common frameworks with well-defined terminology for developing software-based systems from requirements stage to system retirement.
- [6] *Validation of Forensic Tools and Software, A Quick Guide for the Digital Forensic Examiner* by Josh Brunt: Explains that the *Daubert Standard* and NIST's Computer Forensic Tool Testing Project (CFTT) provide requisite guidance for validating forensics tools and software.
- [7] General Testing Methodology for Forensics Tools V1.9, NIST, 11-7-20.
- [8] NIST Computer Forensics Tool Testing Project, <http://www.cftt.nist.gov>; Annex E addresses forensics tools and software.
- [9] *BitStalker: Accurately and Efficiently Monitoring BitTorrent Traffic* by Buer et. al.: The authors conduct experiments, investigate and evaluate the reliability of an active monitoring method for reducing false positives.
- [10] *Open-Source Digital Forensics Tools, The Legal Argument*: by Brian Carrier: Explains that scientific evidence in US court must be reliable and relevant, tool reliability can be tested applying "Daubert".
- [11] *Ninth Circuit Ruling Reminds Copyright Owners that Failing to Allege Enough Facts can cost them!* Jaburg-Wilk, Attorneys at Law, Phoenix, AZ, 10/22/18, discuss the *Cobbler Nevada LLC v. Gonzales*, 901 F.3d 1142 case concluded an infringing IP address, standing alone, does not create a reasonable inference that the subscriber was the infringer, more is required than mere allegation.
- [12] *Daubert v. Merrell Dow Pharmaceuticals Inc.*, 509 U.S. 579 (1993).
- [13] *Challenges and Directions for Monitoring P2P File Sharing Networks*, M. Piatek, T. Kohno and A. Krishnamurthy, University of Washington: authors' experiment using Gnutella and BitTorrent protocols; False positives rates may be elevated by malicious users, malware and buggy software.
- [14] Case No.: 3:16-cv-432-BAS-NLS, USDC (CA), *Malibu Media, LLC v. Doe*, Mar. 28, 2016, Order Granting Plaintiff's Ex Parte Motion ..., <https://casetext.com/case/malibu-media-llc-v-doe-72>.
- [15] Case No. 3:15-cv-0907-AC, USDC (OR), *Dallas Buyers Club v. Huszar, Motion for Summary Judgement*, Feb. 18, 2018, <https://bahnhof.se/filestorage/userfiles/Nyheter%202018/Huszar.pdf>.
- [16] E. Van der Sar, *Torrent Tracking Evidence is Flawed and Unreliable, Alleged Pirate Argues*, Mar 20, 2017, <https://torrentfreak.com/torrent-tracking-evidence-is-flawed-and-unreliable-alleged-pirate-argues-180307/>.
- [17] E. Van der Sar, *Copyright Troll Now Has its Own Piracy Tracking Tool*, Feb 23, 2020 <https://torrentfreak.com/copyright-troll-now-has-its-own-piracy-tracking-tool-200223/>.

# Leveraging Wiring the Winning Organization for Organizational Transformation

**Ron Wilson**

Email address: [ronbwilsonii@gmail.com](mailto:ronbwilsonii@gmail.com)

## 1. Abstract

Problem: Organizations often have processes in place, but they are either not followed or not regularly reviewed and updated. This often causes process issues which lead to frustration, inconsistency and deliverable dates getting missed.

Solution: Leverage the principles found in Wiring the Winning Organization to change how the organization operates to achieve success. Success for each organization will look a bit different since there isn't a one size fits all model. Organizational transformation will not occur all at once. It is a continuous journey with a combination of small and large changes. It will require champions at all levels to help drive change. Each process must be analyzed to determine how it can be most efficient and those processes must be defined and understood by those who are using them in order to make them repeatable. When all the processes are put together, the organization will achieve the objectives it was created for.

## 2. Biography

Ron Wilson is the Software QA Manager at Powin. In his current role Ron manages a remote workforce of software QA engineers across the globe. Ron has over 20 years of experience leading globally distributed quality assurance teams across a variety of industries. He believes in leading people by example with his hands on leadership style. He has an exceptional track record of building software quality organizations from the ground up and is an advocate of developing standard processes to achieve consistent results. Ron leverages metrics to drive organizational transformation and influence change.

Ron has participated in various testing conferences. He has a software testing blog at [www.qarevolution.com](http://www.qarevolution.com) and a YouTube channel called QA Revolution.

Copyright Ron Wilson 7/30/2024 (optional)

## 1 Introduction

Several months ago, I found *Wiring the Winning Organization* (Kim & Spear, 2023) and recommended that my company, Powin, read it to see if we could use some of the principles within the book to improve our processes. Powin is a global leader in the energy storage business. There are several products at Powin which store energy in large batteries to help utility companies take advantage of wind and solar power. As a fun fact, Powin recently powered the Super Bowl in Arizona using renewable energy. This technology is constantly evolving and requires robust testing to ensure the platform is safe and reliable.

Powin grew rapidly and as most startups go, the work had to be done quickly to meet customer priorities and there were limited resources to get the product out the door. Priorities were constantly changing, creating a series of start and stop features, which was further slowing work down. This led to a lot of frustration among the engineering team. In addition, for the most senior members of the organization, they were being torn between supporting production issues and new feature development work which carried the same equal importance in terms of priority.

In my current role, I lead a remote workforce of software QA Engineers across the globe. The software QA Engineers are distributed across several Agile teams. Originally the team had only a few members and they worked together to use best practices based upon their previous experience but it wasn't formally defined and there was a lack of consistency in terms of how things were accomplished. There was a need to standardize the testing processes so they were consistent across the Agile teams.

If we were going to have a chance at winning, something had to change. Change is the only constant in organizations moving forward, and we must do everything we can to keep adjusting to meet the company expectation of delivering software on time with high quality. In order to meet this expectation processes must be analyzed in further detail both at an individual level and collectively as a whole. This paper is a deeper dive into the recommended strategies to build a winning organization to determine which ones will help the organization you work for win. It also includes several case studies that help see those changes in action and will cover Powin's changes to make our organization compete and win daily.

## 2 Organizations

We create organizations for a variety of reasons, but certainly one of them is to accomplish seemingly common but actually audacious undertakings that one person cannot do alone. The goal may be as ambitious as sending a man to the moon or as common as providing a commercial product or service such as running a restaurant, bakery, or hospital (Kim & Spear, 2023, 7). Organizations can be small, with only a few people, or large across multiple continents. Every organization should have a mission or goal. It is important that each individual working within an organization understand its purpose.

It has been proposed that organizations gain competitive advantages largely by seizing opportunities that are unavailable to others. This concept, led by Dr. Michael Porter's "five forces" from his book *Competitive Strategy*, asserts that organizations enjoy unfair returns by having made the playing field unlevel: by locking in customers (so they cannot consider alternative vendors), preventing suppliers from finding other outlets for their wares or barring individuals from offering competitive products and services (Kim & Spear, 2023, 8). Organizations compete against each other and in many situations they are targeting the same customers, using similar materials, and working under the same regulations.

Organizations that create great products can dominate the industry for many years. Here are a few organizations that have beaten their competition significantly.

Toyota has led in design and production in the auto industry for over 50 years. Despite being woefully uncompetitive in the late 1950s, it gained advantages through superior quality and productivity. It built on those leads by cutting in half the time required for major model upgrades, by cutting from weeks to minutes the time to convert plants from one model year to the next, and

by being incredibly fast to introduce whole products and invent whole new technologies (Kim & Spear, 2023, 9).

In 2007, Apple released the groundbreaking iPhone, with only dozens of software developers creating its applications and user interface libraries. The resulting product redefined what consumers expect from mobile devices (Kim & Spear, 2023, 9).

Here are a few examples where established organizations have decided to make changes in their organizational wiring to become more competitive.

In 2002, Amazon struggled to upgrade its e-commerce software, able to make only twenty software changes per year because of the high risk of outages and the difficulty of coordinating across hundreds or thousands of software engineers. In 2014, however, Amazon was making some 136,000 deployments every day, quickly and safely (Kim & Spear, 2023, 10).

In manufacturing microprocessors, the differences between the leaders and the rest are huge in terms of throughput times, quality, yield, and sustained product variety within a single plant. What's encouraging is that such performance is replicable. One plant cut its throughput times by two-thirds, increased yield, reduced scrap, and otherwise made it far easier for engineers and technicians to use the sophisticated capital equipment they had (Kim & Spear, 2023, 10).

These are just a few examples of organizations which have been able to make changes in processes which have resulted in significant increases in quality, reliability, and profit. These changes did not happen overnight but took time. Amazon for example, took 12 years to get to the point where they were able to scale their deployments with quality. I am confident throughout that timeframe that there were many failures which caused them to pause and reassess before being able to move in a positive direction.

## 2.1 The Three Layers of Value

Organizations work together to accomplish a series of goals. To achieve outcomes, they must collaborate and work together to solve problems. Collaboration isn't optional but is something that is required for all organizations and often one of the greatest obstacles in achieving success. While each part of the organization is working towards the ultimate goal there are often competing priorities between one department and another. For example, if the production support team needs the expertise of a team member who is working on new feature development to solve a difficult defect, the team member will need to pause the new feature development work to help the other team with a production defect.

In the example above, the software engineer needed to solve a complex defect. Let's take a look at the problem solving techniques available which will help to solve the problem.

- **Layer 1:** contains the technical objects. These include technical, scientific, and engineering objects that people are trying to analyze, create or modify (Kim & Spear, 2023, 13). The software engineer will be analyzing and modifying software code which is technical in nature.
- **Layer 2:** contains the tools and instrumentation. These are the scientific, technical, or engineered tools and instrumentation through which people work on Layer 1 objects. The software engineer will need to use the following technical tools which include Jira, Confluence, AWS, Bitbucket, Splunk, and Eclipse and to help solve the production defect (Kim & Spear, 2023, 14).
- **Layer 3:** contains social circuitry. This is the overlay of processes, procedures, norms, and routines, the means by which individual efforts are expressed and integrated through collaboration toward a common purpose. This is the "socio" part of a socio technical system. Processes and collaboration with other members of the organization occur. (Kim & Spear, 2023, 14). The software engineer will

need to use the established processes of the organization and make sure those are followed. One part of the software development process is to have a peer perform a code review to ensure there aren't formatting or potentially other defects in the code before it is merged into the build, retested, and deployed into production.

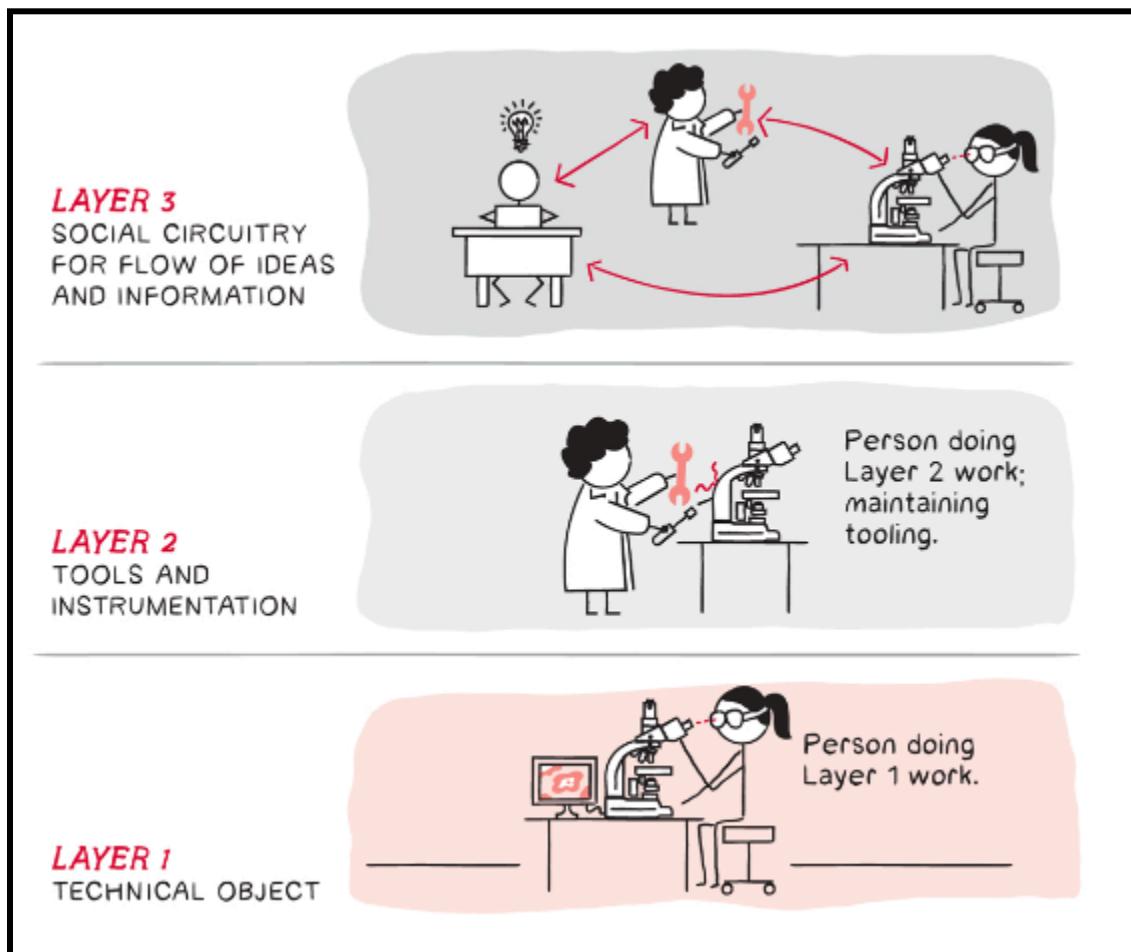


Figure 1: The Three Layers (Kim & Spear, 2023, 16)

These three layers help redesign problems to make them more accessible and faster. In Layers 1 and 2, people can focus on solving problems. The software engineer resolved a technical defect in the code by making the appropriate code changes using the software tools available. If the software engineer doesn't have the proper information or the right tools to help solve the problem effectively, then the software engineer is going to have to spend a lot of extra time working within Layer 3. This will be especially difficult if the organization doesn't have effective processes as it relates to things such as detailed requirements, code reviews, and branching strategies. The software engineer will have to over-compensate and ultimately not deliver the correct resolution to the defect with high quality in a timely manner. This will cause frustration for both the software engineer and the customer.

## 2.2 Danger Zones versus Winning Zones

Leaders must manage complex problems at Layer 3. This helps determine if the organization can achieve terrible or great outcomes. This social circuitry makes resolving problems or generating new ideas easy or difficult. In the danger zone, conditions make solving problems or developing new ideas difficult. In the winning zone, conditions are more straightforward for solving or generating new ideas.

Here are some examples of the *Danger Zone* versus *Winning Zone*.

<b>DIMENSIONS</b>	<b>DANGER ZONE</b>	<b>WINNING ZONE</b>
<b>Nature of problems.</b>	⚠ Complex problems with many highly intertwined factors.	✓ Simplified problems that are well bounded, have fewer factors, and can be addressed by smaller teams.
<b>Hazards and risks.</b>	⚠ Dangerous and risky.	✓ Less hazardous and less costly failures.
<b>Speed of environment in which we're trying to solve problems.</b>	⚠ Fast moving and not controllable.	✓ Slower moving with the opportunity to control pace and introduce pauses.
<b>Opportunities to learn by experience or experimentation.</b>	⚠ Experiences are singular or "one-off" so feedback may be missing and learning loops may not exist.	✓ Experiences can be repeated to gain experiential and experimental learning, and knowledge can be captured for recurring use.
<b>Clarity about where and when to focus our problem-solving efforts.</b>	⚠ It is not obvious that problems are even occurring, so they get neither contained nor resolved.	✓ It is obvious when problems are occurring, so attention is given to containing and solving them; and it's obvious whether the problems have been contained and resolved or not.

**Figure 2: Danger Zone vs. Winning Zone (Kim & Spear, 2023, 16)**

In the *danger zone*, problems are complex, with many factors affecting the system at once, and their relationships are highly intertwined. Hazards are many and severe, risks of failure are high, and costs of failure can be catastrophic. Systems in the *danger zone* are difficult to control, and there are limited, if any, opportunities to repeat experiences, so feedback-based learning is difficult if not outright impossible. These problems are often intertwined, which makes it extremely difficult to manage. The risk of failure is high, and the cost of failure can be catastrophic (Kim & Spear, 2023, 17). At Powin, we struggled to move from a start-up culture with little documentation and standardized processes. Knowledge was limited to a select few and not distributed across the organization. Those key resources were constantly overworked and burnt out and the organization was stuck based upon the capacity of a few key players. Priority was getting the software development done and not following processes or sharing knowledge. Without a deliberate change, the same cycle would continue.

In contrast, leaders enable much more advantageous conditions in the *winning zone*. Problems have been reframed so they are simpler to address. The hazards and risks have been reduced so failures are less costly, especially during design, development, testing, and practice. Problem-solving has been shifted into slower-moving situations, where the pace of experiences can be better controlled. Opportunities to learn by experience or experimentation are increased to allow for more iteration. And finally, there is more clarity about where and when to focus problem-solving efforts, because it is obvious when problems are occurring, so attention is given to containing and solving them (Kim & Spear, 2023, 17). At Powin, we made a deliberate attempt to break the cycle. Sure, we still have issues and sometimes revert back to old habits, but we are much more consistent in terms of our process. Instead of pushing through features to meet the schedule, we put things on hold, get the problems resolved, and

then resume feature development. This helps to release a feature with higher quality versus delivering one that will leave our customers unhappy.

### 2.3 3 Ways to Solve Problems

Three mechanisms are available to solve problems moving from the *danger zone* to the *winning zone*.

1. **Slowification:** makes it easier to solve problems by pulling problem solving out of the fast-paced and often unforgiving realm of performance. Instead, solve problems this in the more controllable, forgiving, lower-cost, less-demanding, and repeatable realms of planning and practice. This shifting of Layer 3 problem-solving into planning and practice allows people to engage in iterative, reflective, experiential, and experimentally-informed reasoning rather than having to constantly react with whatever habits, routines, and legacy approaches have already been ingrained (Kim & Spear, 2023, 18)
2. **Simplification:** makes the problems themselves easier to solve by reshaping them. Large problems are deliberately broken down into smaller, simpler ones through a combination of three techniques: incrementalization, modularization, and linearization. By doing so, we partition complex problems with many interacting factors into many smaller problems. These problems have fewer interacting factors, making them easier to solve. Furthermore, Layer 1 (technical object) problem-solving can be done in parallel, with less need for Layer 3 coordination, increasing independence of action (Kim & Spear, 2023, 18)
3. **Amplification:** makes it obvious there are problems, and makes it clear whether those problems have been seen and solved. Mechanisms are built into Layer 3 (social circuitry) to amplify that little things are amiss, drawing attention to them early and often. This focuses attention on containing and resolving small and local glitches before they have a chance to become large and systematically disruptive (Kim & Spear, 2023, 18)

The organization can leverage using one or all of these options for each problem—slow things down to make things easier to solve. Break down significant problems into smaller ones. Amplify when there is a glaring problem that needs to be solved. These techniques will help ensure success and move the organization to optimal performance.

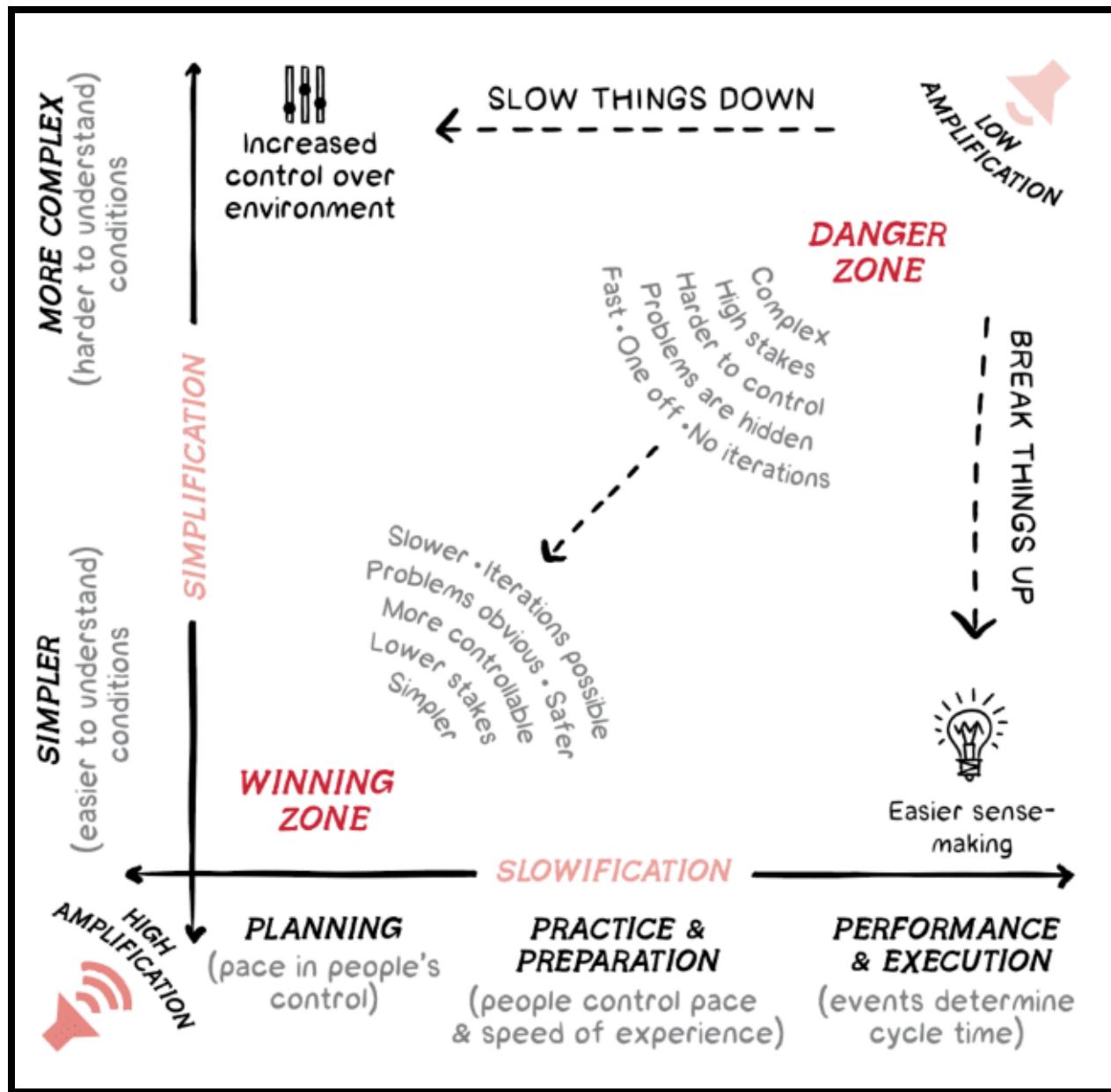


Figure 3: Moving from *Danger Zone* to *Winning Zone* (Kim & Spear, 2023,52)

Based upon the figure above, in simplification, complex problems are broken down into simpler problems in order for the larger problem to be solved. For example, if there is a large complex feature which needs to be developed those components are broken down into multiple phases. In slowification, when things are slowed down, it allows opportunities to plan which allows more control over the problems which are encountered. For example, if a feature is ready for testing, but when testing begins there are several bugs which are encountered due to design issues, that feature testing is put on hold and sent back to development so those bugs can be resolved before testing resumes. In amplification, when there are problems attention needs to be given to those problems in order for them to be solved in a timely manner. For example, if the requirements of a given feature don't provide enough detail for the development team, that problem is escalated to project management in order for those requirements to be further defined, so the development team has the right amount of information needed to proceed with feature development.

## 3 Slowification

Slowification is the first of the three mechanisms that move organizations out of the *danger zone* and into the *winning zone*. There are many maxims that convey a similar idea: go slow to go fast, pause to sharpen the ax, and so forth. Yet there's no single English word that conveys the duality of the idea that going faster often depends on going slower. In the absence of such a word, we opted to make one up: slowification (Kim & Spear, 2023, 70).

As an example, there was a problem where a software feature needed to be developed quickly for one of our customers. The team decided to rush through the requirements, design, and development phase. When it was handed over for testing, multiple bugs were discovered within a few hours. Instead of continuing to press through, the team decided to put testing on hold and go back and create acceptance criteria, build unit tests and execute those before it moved back into testing. While this effort slowed things down initially, it reduced the amount of defects identified during testing and accelerated the testing phase. In addition, our internal product team and the customer were happy with this feature. This allowed them to gain more confidence in our product and know that future feature requests will have higher quality which will meet their needs.

### 3.1 Fast Thinking vs. Slow Thinking

There are two modes of thinking: fast thinking and slow thinking. Dr. Daniel Kahneman, 2002 Nobel Laureate in Economic Sciences, and his colleague, the late Dr. Amos Tversky, distinguished between two thinking modes: System 1 (or fast thinking) and System 2 (or slow thinking) (Kim & Spear, 2023, 71).

I have given this much thought and practiced it several times and it really makes a lot of sense. I find myself often rushing to solve a problem and on a typical day I am usually faced with at least 20 problems that I need to solve. There are some problems where I can apply fast thinking and resolve those problems based upon my years of experience and familiarity within my work environment. There are times when those quick decisions backfire because I didn't take enough time to analyze and come up with the proper solution. Fast thinking can definitely cause some major problems. For example, if I would have performed slow thinking to analyze the bug that was found during testing, I would have determined that it should have been a release blocker and should have been fixed before the code was deployed to production. If this change would have deployed into production, our customers would not have been able to move power using our platform.

When we are thinking quickly, we are using only what we already know. Fast thinking doesn't encourage or allow time to improve our thinking. This muscle memory works in situations for which we've practiced, but it fails us in situations that are unfamiliar (Kim & Spear, 2023, 71).

Slow thinking is used most effectively when the proper amount of time is spent to solve a problem. If it is a new problem we are facing, we don't have the luxury of relying on our experience because it is new. For example, if you are going to a new city to meet a customer, it is a place you are unfamiliar with. While you have made the car, hotel, and plane reservation and even had video conferences with the customer, it will take you extra time to get familiar with your surroundings and be comfortable in meeting the customer in person for the first time. The next time you make the trip you can apply fast thinking because you have already had the experience and it will be much easier on the second visit.

	<b>STRENGTHS</b>	<b>LIMITATIONS</b>
<b>Fast thinking</b>	<b>Speed:</b> Heuristics give us quick, reliable answers to familiar problems and situations.	<b>Inaccuracy:</b> Heuristics and their attendant biases may give us inaccurate answers to unfamiliar problems or situations that are framed poorly.
<b>Slow thinking</b>	<b>Flexibility:</b> Allows us to improve our understanding of familiar situations or add to our understanding of new ones.	<b>Slowness:</b> Requires time, patience, and openness of mind that we may be lacking in the moment.

**Figure 4:** Advantages and Disadvantages of Fast and Slow Thinking (Kim & Spear, 2023, 71)

### 3.2 Cognitive Bias

We are all prone to cognitive biases. Biases show up everywhere, including how we perceive individuals (Kim & Spear, 2023, 73). As an example, I once hired a contractor who previously worked at Google. I had interviewed the candidate and he interviewed extremely well and I was biased to believe that he would be a great fit. When he joined, he was arrogant, didn't ask for help and struggled to learn the product and wasn't a team player so I ended up having to let him go. Reading this book has helped remind me that I have cognitive bias and that I need to be more careful when making decisions. If I see a potential candidate from a leading technology company, I will perform more due diligence with that candidate to gain a better understanding of their ability to collaborate with other members of the team and their desire to learn new products. I will also involve more team members in the decision making process to reduce my cognitive bias with this particular situation.

### 3.3 The 3 P's

Because of the limitations of fast thinking, in many situations we want to employ slow thinking. We want to create opportunities where we can be more deliberate, contemplative, self-reflective, and inquisitive. In this unrushed, lower-risk environment, we can deliberately challenge our pre-existing mental models and update them based upon additional experiences and experimentation. This is essential if we are going to get to new and useful outcomes and discover great things (Kim & Spear, 2023, 71).

As a leader of a team, it is important that I spend time thinking about standard processes which the team needs to follow. I have developed processes in fast thinking mode and rushed to get them out to the team, and often found those processes have gaps and it requires additional changes to make them work.

The *planning environment* (e.g., design or development) is the slowest-moving, lowest-cost, safest environment in which to develop and test ideas. In the context of this paper ideas include processes or software feature development. Here, ideas are literally or figuratively words and drawings penciled on paper, so the cost of expressing and capturing them is low. Ideas can be tested many times (e.g., through mock-ups, simulations, thought experiments). The low cost also allows running iterative experiments toward ever-improving solutions. This is the fastest, quickest, easiest environment in which to find flaws in thinking before they become flaws in doing (Kim & Spear, 2023, 75).

I am a firm believer that Agile planning in software development is essential. According to Mike Cohn, The key to successful agile planning is to recognize that it is an ongoing process, not a one-time event. Planning is important because it helps the team focus on the highest-priority tasks and ensures that everyone is aligned on the goals and objectives of the project (Cohn, 2005). Without proper Agile planning, the execution and delivery are going to have issues. As an example, when a feature was delivered, there was a rush to jump straight to building test cases in preparation for the delivery of code

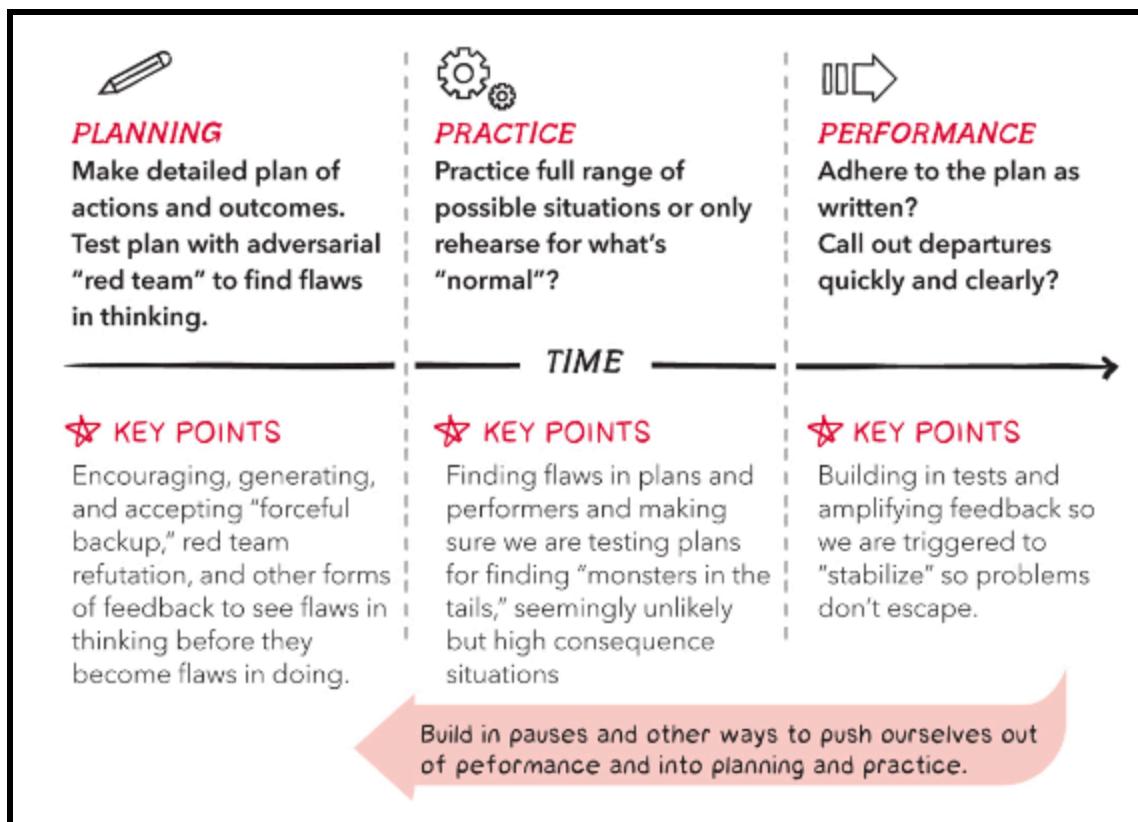
so the test cases could be executed. While the functional tester was greatly constrained by time and pressured to test quickly, it ultimately resulted in features having poor quality. I decided the team must focus more energy with test planning and created a process to require test plans for every feature. In addition, I recognized that test planning should be a collaborative activity across Architects, Developers, and Product Management. By including other teams in the process, it helped to gain additional input and make the test plans more robust. This process change resulted in improved quality reducing the number of defects in production.

The *practice environment* (e.g., pre production, testing, offline problem-solving) is a more demanding environment than planning because ideas are being put into action. However, actions are taken in a safer and more controllable environment. In practice, we can control the pace and the (incremental) complexity of what is occurring so we are not overwhelmed. We can also build multiple learning cycles into the experience, which allows us to keep adjusting what we are doing and how we are doing it (Kim & Spear, 2023, 76).

It is critical to have multiple test environments in software development since each environment will serve a different purpose. As an example, in my current organization, we have 5 environments. The development environment is where all new feature development and testing of those features take place. This environment has the least amount of restrictions and development and testing is done on the latest code branch. Code builds are deployed multiple times per day. The pre-production environment is a bit more restrictive in terms of user permissions and the code delivery is controlled. Code builds can only be deployed by Quality Engineering. This environment is used for final testing of the features and regression. The staging environment is used to deploy the approved code and run some basic sanity tests to ensure there are no issues. If there are issues, the code is rolled back and fixes are retested before it is deployed back into staging. The staging environment is further restricted in terms of users. The production environment is the most restrictive environment and code is deployed only with approved builds that have moved through all the other environments. We have one final environment which is our break fix environment. This environment has a copy of the production code and fixes are deployed into that environment when we need to deliver a patch outside of our normal release schedule. These environments were built over a period of time and helped to reduce the amount of conflicts we had in using the environments and helped to improve the quality of the code that was delivered to production. While the customer will not be aware these changes are being made, in the end they will be using a higher quality product that operates more efficiently and is safer to use.

The *performance environment* (e.g., operations, execution) is the most unforgiving environment. It controls the pace of the experience, forcing us to depend almost exclusively on already-developed routines, skills, and habits. We likely have limited ability to redo or otherwise correct our actions in this environment. Performance is the last place we can learn, but only if we recognize that our situation is different from expected or needed. This is also what enables us to swarm problems, limit their duration, contain their spread, and solve them. It also enables us to share and systematize what we've learned so the problem doesn't reoccur. (Kim & Spear, 2023, 77).

Production environments should be restrictive and have the proper processes in place to ensure the software code is the highest quality and can perform repeatable processes over and over again. Our production environment is restricted by who has access to the environment and what they are allowed to do within the environment. Code deployments can only be done by a few members of the software engineering organization. When there are issues in production, there is an escalation process in place to quickly resolve those issues. In addition, we have a root cause analysis process which helps to ensure critical problems are quickly resolved and limit the possibility they will occur again.



**Figure 5:** The Three Ps: Planning, Practice, and Performance (Kim & Spear, 2023,75)

The figure above illustrates some important points to keep in mind with these principles. As it relates to planning, when a functional tester is going to test a new feature they must create a test plan as a part of the process. Instead of creating the test plan for their own use, they should collaborate with Architects, Developers and Product Management to build a robust test plan. This will result in a higher quality product delivered to the end customer. As it relates to practice, it is important to recognize that there might be edge cases or other features which have dependencies that should be included in the test plan to confirm both the edge cases and dependent features work as intended. As it relates to performance, it is important that the functional tester follow the test plan and identify issues as they occur. Recently a functional tester collaboratively worked with Architects, Developers, and Product Management to build a test plan for a new feature. When the code was delivered, there were other safety features which prevented testing of several of the test cases identified in the test plan. The functional tester amplified the issue so the stakeholders would be aware that the test plan would change based upon the unknown dependency. The functional tester proceeded to complete the testing and the feature was successfully delivered to the customer.

### 3.4 Capture Knowledge

Regardless of which phase we find ourselves in—planning, practice, or performance—how often and well do we capture lessons learned in a way that is easy to access? And how well do we create opportunities for those lessons to be learned so they can be incorporated into future planning and used in future practice? As we think through opportunities in our own work, consider that knowledge capture has been a key mechanism leading to betterment through human history, taking full advantage of writing, imaging, modeling, and so forth (Kim & Spear, 2023, 122).

In my current organization, there are a few ways we capture and gain knowledge.

1. Online Training: There are online training courses every employee is required to take. This helps to ensure that all employees learn about safety and human resources policies. Without having this knowledge captured in training courses, it would be impossible for the employees to know what policies are in place and violations would constantly occur.
2. Centralized Online Content: We use web pages to document product requirements and software development processes. These pages are updated when the information changes.
3. Lunch and Learn: Lunch and learn sessions are an effective way to encourage continuous learning and knowledge sharing within agile teams. These informal gatherings allow team members to share insights, best practices, and experiences in a relaxed setting, fostering a culture of collaboration and ongoing education (Elssamadisy, 2008). In our software development organization we have bi-weekly meetings. For the new employees, this is one of the items they must complete as a part of the onboarding process. Knowledge sharing is extremely important since we have grown rapidly over the past several years and we recognize the importance of distributing the knowledge from a few key resources to the whole team.
4. Sprint Planning: Regular planning sessions, such as sprint planning, are critical in agile as they allow teams to adjust their priorities, set realistic goals, and ensure that everyone is aligned with the project objectives. These sessions provide a structured opportunity to plan the upcoming work while taking into account the feedback and learnings from the previous iteration (Schwaber, 2004). For the software development organization, we hold a monthly meeting to review innovation projects, team retrospectives, and review features planned for the upcoming release.
5. Daily Standup: The daily stand-up is a meeting where each team member shares what they did the day before, what they will do today, and any impediments they are facing. This practice not only keeps everyone informed but also fosters a culture of knowledge sharing, where the team can collectively address challenges and synchronize their efforts (Sutherland & Sutherland, 2014)
6. Stakeholder Demos: The purpose of the stakeholder demo is not just to show what has been accomplished, but to share knowledge, gather feedback, and ensure alignment between the team and stakeholders. It's an opportunity to validate the work done and to foster collaboration and transparency (Rubin, 2013).

## 4 Simplification

Simplification makes it easier to solve problems by changing the conditions to solve the problem thus making them easier to solve. Simplification makes the problems themselves easier to solve. For example, by limiting the number of active projects and streamlining tasks, the team can reduce context switching, decrease errors, and accelerate delivery times. This approach to simplification ensures that the team can focus on completing one task at a time, leading to more predictable outcomes and higher quality in the software development process (Kim & Spafford, 2013).

Simplification is achieved through three techniques—incrementalization, modularization, and linearization. In short, simplification breaks up situations that are big, complex, convoluted, integrated, or highly intertwined and makes them more manageable because they are smaller, contain fewer departures from what is already known, and are easier to understand in their construction (Kim & Spear, 2023, 130).

Moreover, simplification offers another advantage: breaking large, complex problems into smaller, easier pieces means more problem-solving can occur simultaneously (in parallel) with less coordination required. The net result is making problems easier to solve and solving more of them at once. This ability to solve more problems independently and in parallel is a consequence of all three simplification techniques partitioning large systems into smaller, coherent pieces (Kim & Spear, 2023, 130).

In our product, we have complex features that are developed. The features typically span across multiple teams and will involve hardware, firmware, and many software code repositories. The feature must be broken down into smaller components in order to build the feature and ultimately deliver it to the end customer. Our initial strategy was to have a single feature for each team. Some of these features would be large and could include up to 40 sub-features. It would typically take 2-3 months to deliver the feature and we would wait until that feature was fully complete and deploy the completed feature into production. After reading this book, we decided to change our process. Instead of having a single large feature, we Excerpt from PNSQC Proceedings

would break down the feature into several smaller features. In addition instead of waiting for the full feature to complete, we would deliver the work incrementally into production. This process change did several things.

1. Breaking the feature into smaller pieces, allowed those pieces to be delivered incrementally.
2. It improved the overall morale of the team because they could see progress with the smaller epics getting completed and delivered.
3. It allowed delivery of partial features to our customers.
4. If the partial feature delivered wasn't meeting customer expectations, it allowed the opportunity to resolve those issues so when the full feature was delivered, it would result in greater customer satisfaction.

Simplification uses three techniques:

## **4.1 Incrementalization**

Incrementalization partitions what is novel (which needs to be tested) from what is known (which is already validated) into their own self-contained, coherent units and adds to the novelty in many smaller increments rather than in a few large attempts. The benefit is that we iterate and test changes on fewer factors and on a smaller portion of our system more quickly and safely (Kim & Spear, 2023, 132).

From the example above, instead of delivering the full feature after several months of development, It was delivered in monthly increments. This encouraged more frequent feedback which helped to increase customer satisfaction when the final part of the feature was delivered.

## **4.2 Modularization**

Modularization partitions a large, integrated system that is unwieldy in size, complexity, or intertwined relationships into smaller, simpler, more numerous coherent pieces. These coherent modules are less coupled to each other because they are connected through only a few well-defined and stable interfaces. The benefit is that small teams gain independence of action, enabling them to work and experiment on more manageable parts of the problem in parallel and more quickly and safely, with lower costs of coordination (Kim & Spear, 2023, 132).

After reading the book, we decided to change our process to allow various components of code to be delivered independently into production. In order to make the change we needed to decouple the components and validate that those components could operate independently. This allowed certain features, in our case the UI and Data API components to be delivered more frequently into production. This resulted in higher customer satisfaction.

## **4.3 Linearization**

Linearization partitions operations that are complex and share resources to accomplish multiple objectives into independent (decoupled) and coherent workflows. Each is focused on one or a few objectives that can happen in parallel. (Kim & Spear, 2023, 132).

In software testing there are several activities that need to take place when testing a new feature. For the most part the activities are completed in a sequential workflow. The work tasks include: test planning, test execution, defect retesting, and regression. In our organization we have a functional team and an automation team. For the test cases that need to be executed frequently, especially after each release, we have a process in place to identify those test cases and assign them to the automation team to build and have them executed without human intervention. This allows those tests to run faster than if someone on the functional team executed them manually. Both teams are able to focus on their respective tasks independently and ultimately achieve the objective using the linearization technique.

## 5 Simplification Implications for Leadership

How you lead an organization that has been simplified—through some combination of incrementalization, modularization, and linearization—differs massively from how you lead one that has not been simplified (Kim & Spear, 2023, 213).

Understanding this concept has helped me change how I lead in focusing on smaller components such as a problem with a current process or an identification and implementation of a new process. This mindset change has enabled improvement of the standard set of procedures which the organization operates. Once a series of problems are solved or processes are simplified, I also reflect on those changes, to see what additional improvements can be made.

### 5.1 Incrementalization in Leadership

	<i>ALL-AT-ONCE LEADERSHIP</i>	<i>INCREMENTAL LEADERSHIP</i>
<b>Attention</b>	Diffused over many things, simultaneously.	Focused on what is novel but not yet functional or reliable.
<b>Leadership priorities</b>	Giant leap.	Many small steps.
<b>Leadership challenges</b>	Keeping pace with systems scale, scope, complexity, and speed.	Maintaining channels of communication and mechanisms for knowledge sharing and exchange.
<b>Key responsibility</b>	Determining who should be doing what, for what reason, in what fashion.  This is by necessity fast-paced, complex, and highly detailed.	Partitioning novel from validated and ensuring experiments are being conducted rigorously and frequently.
<b>Problem-solving</b>	Forced into a few cycles of complex experience and experimentation; difficult sense-making with few learning-loop iterations.	Allowed more cycles of experimentation with easier sense-making and gradual introduction of scale, scope, and complexity.

**Figure 6:** Leadership Challenges with All-At-Once vs. Incremental Approaches (Kim & Spear, 2023, 213)

In the figure above, it provides a clear illustration of the differences in leadership style between all at once and incremental leadership. The all at once leader is likely going to constantly be stressed and is going to have difficulty managing everything at once, and in the end will accomplish less than if the incremental leadership techniques were leveraged. The incremental leader leverages others to gain knowledge and encourages sharing of that knowledge. Focused attention on conducting a series of experiments helps to understand more about the problem and ultimately come up with a better solution.

## 5.2 Modularization in Leadership

	<i>TOP-DOWN LEADERSHIP</i>	<i>CENTER-OUT LEADERSHIP</i>
<b>Data</b>	Centralized.	Distributed.
<b>Decision rights</b>	Centralized.	Distributed (but bounded).
<b>Solutions</b>	Homogeneous	Heterogeneous.
<b>Leadership priorities</b>	Coordination and control.	Facilitation, communication, and synthesis.
<b>Leadership challenges</b>	Keeping pace with systems scale, scope, complexity, and speed.	Creating and maintaining channels of communication and mechanisms for knowledge sharing.
<b>Key responsibility</b>	Determining who should be doing what, for what reason, in what fashion. This is by necessity fast-paced, complex, and highly detailed.	Creating mechanisms by which people can arrive at their own solutions and have local discovery synthesized into system solutions.
<b>Mode of problem-solving</b>	Leaders are forced into fast-thinking habits, routines, and impulses to be responsive to demands from operating units.	Leaders are able to maintain deliberative, slow-thinking approaches of designing, assessing, and improving the mechanisms they've created for data sharing and knowledge synthesis.

**Figure 7:** Comparing Top-Down vs. Center-Out Leadership (Kim & Spear, 2023, 215)

The figure above illustrates the differences between top-down leadership and center-out leadership. The two styles are very different. I prefer having a leader who practices center-out leadership and therefore I try to model that behavior. In my opinion, it is very difficult to manage using the top-down leadership style and it usually results in making poor decisions. The reason poor decisions are made is this style only considers a single perspective and does not encourage getting a different perspective that would result in a better decision. My preference is to share knowledge, maintain communication, and allow team members to come up with solutions based upon their own discovery. There are times however, where I might have to use top-down leadership. For example, when there is a release deadline and if I feel there is potential for the testing to not finish on time, I will provide direction on what needs to be accomplished.

## 5.3 Linearization in Leadership

Just as modularization and incrementalization change the nature of leadership, so does linearization. Shifting to linearized workflows partitions the entire enterprise into distinct pieces. For example, instead of having a single software development team focus on both new feature development and production support issues, split the team into two different groups so each team is able to have a single priority. That has benefits, like shifting from top-down to center-out management. Leaders have to spend less time doing frenetic, impulsive, fast-thinking work. Instead, they can be more deliberative, first figuring out the partitioning that allows for linear flows. That's deliberative slow thinking, a more productive use of people's minds. Then, they can continue in the slow-thinking, deliberative, productive mode by solving the engineering (technical) problems that affect safety, cycle time, quality, and yield (Kim & Spear, 2023, 216).

As a leader, I need to understand the problems that my team has on a daily basis. This allows me to help remove the roadblocks so they can efficiently complete the tasks they are working on and be able to move to the next task. One particular challenge we have had is the developer having a different setups on their computers versus how the computer is set up for the tester. We decided to invest time in building a separate team that would be focused on building Docker images that can be shared between the developer and tester in order to ensure consistency and be able to reproduce the issues regardless of the computer that is used. This process change will help to resolve the issues faster and help ensure consistency in how the software is configured. While this is a work in progress, I am confident this change will reduce the frustrations and allow the developer and tester to focus on higher value work that is completed.

## 6 Amplification

*Amplification* is the act of calling out problems loudly and consistently enough so help is triggered to swarm them. Once the problems are swarmed, they are contained so they neither endure locally nor spread systemically. Then, they are investigated to determine their causes and create corrective actions that prevent recurrence. This requires that the signal of a problem is successfully generated, transmitted, received, and then reacted to (Kim & Spear, 2023, 233).

As a leader of a software quality engineering team, this is a technique I use often. Even before I read the book, I would frequently call out blocking issues and escalate those issues to project management or development. Recently, we had a major release for our largest customer and we were faced with a very tight deadline. While I had called out issues which were blocking testing on a daily basis in the daily standup, the issues were not getting solved in a timely manner. I decided to create a slack channel specifically for that project and include the leadership team. The leader of the organization recognized the importance of removing those blocking issues and encouraged his leadership to quickly work to resolve those issues. By using the *amplification* technique, we were able to complete delivery of that critical project.

## 7 Conclusion

Organizational Transformation is a journey. It starts with a single person's desire to transform the organization into something different. One successful change led to another change, creating a series of changes. *Wiring the Winning Organization* provides various principles and techniques to influence change. Some changes will fail miserably, and that is perfectly acceptable. I highly encourage failure for the organization to have the experience to learn from that failure. This takes an extreme amount of courage. Let's face it, nobody wants to fail. It will require you as the change agent to lead by example and be willing to demonstrate that failure is acceptable. Once the failure turns into a successful change, others will be willing to fail to help the organization change.

Organizational Transformation requires change agents who are willing to roll up their sleeves and champion the change. It involves change agents at every layer of the organization. I desire that this paper will fuel a passion inside you to become a change agent, take just a few of these principles, and challenge the status quo at your organization to influence real change. That is the only way organizations can improve and win against the competition. Do not be afraid to speak out. Your voice is valuable. Go out and compete and win!

## 8 Next Steps

Here are the next steps I recommend in starting your journey towards Organizational Transformation.

1. Read *Wiring the Winning Organization*.
2. Recommend a few of the key leaders within your organization read the book and ask them if they would like regular meetings to review the book chapter by chapter.
3. Identify ways to make a few small changes within the organization as a proof of concept.

Once you have done these three steps and determined the concept will work within your organization, have a kick-off meeting to review the changes and encourage everyone to join the Organizational Transformation Journey.

## References

- Cohn, Mike. 2005. *Agile estimating and planning*. Prentice Hall.
- Eissamadisy, A. 2008. *Agile adoption patterns: A roadmap to organizational success*. Addison-Wesley Professional.
- Kim, Gene, and Behr, Kevin, & Spafford, George. 2013. *The Phoenix Project: A novel about IT, DevOps, and helping your business win*. IT Revolution Press.
- Kim, Gene, and Steven J. Spear. 2023. *Wiring the Winning Organization*. IT Revolution Press.
- Rubin, K. S. 2013. *Essential Scrum: A practical guide to the most popular agile process*. Addison-Wesley.
- Schwaber, K. 2004. *Agile project management with Scrum*. Microsoft Press.
- Sutherland, J., & Sutherland, J. J. 2014. *Scrum: The art of doing twice the work in half the time*. Crown Business.

# Turning Testers into Superheroes

**Jonathon Wright**

me@jonathon.ai

## 1. Abstract

In this paper I will share my experience journey from a novice tester in the 1990s to a quality assurance professional, exploring the evolution of testing approaches over decades. The author highlights the redundancy and inefficiency of repetitive manual test design. However, the advent of Generative AI and Large Language Models (LLMs) marks a transformative era in software testing. While generative AI offers vast potential, its current limitations include a lack of domain-specific knowledge and issues with data privacy. Despite these challenges, offline generative AI can leverage proprietary knowledge without risking data security, enhancing automated testing through model-based approaches.

The paper argues for the value of testing as a critical business enabler, emphasizing that QA should be perceived as adding value rather than a cost. The shift from rigid IT structures to user-centric designs has made testing more complex, necessitating faster and cheaper solutions without compromising quality. Generative AI, when combined with human expertise, promises to overcome these challenges by integrating testing into the development phase, reducing cognitive biases, and accelerating test design processes. The author contends that AI complements human testers, enhancing productivity and enabling a better work-life balance. Ultimately, the paper envisions a future where generative AI transforms testers into superheroes, facilitating collaboration, and significantly elevating the role and perception of QA in business environments.

## 2. Biography

*Jonathon Wright is a strategic thought leader and distinguished technology evangelist. He specializes in emerging technologies, innovation, and automation, and has more than 25 years of international commercial experience within global organizations. Jonathon combines his extensive practical experience and leadership with insights into real-world adoption of Cognitive Engineering (Generative AI). In his spare time, he is a member of Harvard Business Council, A.I. Alliance for the European Commission, chair of the review committee for the ISO-IEC 29119 part 8 “Model-Based Testing” and part 11 for the “Testing of A.I. based systems” for the British Computer Society (BCS SIGiST). Jonathon also hosts the QA lead (based in Canada) and the author of several award-winning books (2010 – 2022) the latest with Rex Black on ‘AI for Testing’.*

Copyright Jonathon Wright, 2024

## 1 Introduction

The landscape of software testing has undergone significant transformations over the past few decades, driven by technological advancements and evolving business needs. In this paper, chronicles my journey in software testing, offering insights into the challenges, developments, and future prospects of the software testing domain. This paper begins with a personal narrative from the 1990s, a time when the author, a novice tester, embarked on a summer job at a large telecommunications company. The task at hand was to manually test a hefty book of test requirements and test cases. This introduction serves to set the stage for a broader discussion on the evolution of testing practices, the role of automation, and the emerging influence of Generative AI in this field.

During the initial phase of the author's career, software testing was a tedious, manual process. The six weeks spent running thousands of tests without encountering a single failure illustrated a fundamental inefficiency: repetitive manual testing that failed to reveal new insights or add value. This realization sparked a determination to automate these processes. Automation promised not only to expedite the testing process but also to liberate testers from monotonous tasks, allowing them to focus on more value-adding activities. This shift towards automation marked a significant milestone in the author's journey and in the broader field of software testing.

The narrative then transitions to the present, highlighting the advent of Generative AI and Large Language Models (LLMs) as a revolutionary force in testing. Over the past year, these technologies have demonstrated extraordinary potential in various domains, from code generation to answering complex queries. However, their application in software testing is not without limitations. Generative AI currently lacks domain-specific knowledge and cannot verify the validity of its responses without human expertise. Furthermore, concerns about data privacy have emerged, particularly with instances of proprietary code being inadvertently used as training data for AI models. Despite these challenges, the author posits that offline Generative AI, which operates within secure environments, can effectively utilize proprietary knowledge, optimizing and prioritizing test execution through model-based approaches.

The core argument presented in this paper is that testing should be viewed as a critical business enabler rather than a cost. The IT landscape has shifted from rigid, workflow-centric applications to user-centric designs, significantly complicating the testing process. Organizations today demand faster release cadences, better quality, and lower costs—a trio of requirements that traditional testing struggles to meet simultaneously. Agile and DevOps methodologies have introduced concepts like chaos engineering, yet testing often lags behind these rapid development cycles. The author asserts that integrating Generative AI with human expertise can address these challenges, enhancing collaboration, reducing cognitive biases, and accelerating the testing process.

This paper explores the transformative potential of Generative AI in software testing, arguing that it can turn testers into superheroes by handling the heavy lifting of test design and execution. While AI can manage vast amounts of data and generate comprehensive test scenarios, human testers bring essential wisdom and contextual understanding to the process. By leveraging both AI and human strengths, the testing function can significantly enhance its value, ensuring better quality software delivered faster and at lower costs. The introduction concludes by envisioning a future where AI and human collaboration redefine the role and perception of quality engineering in business environments.

## 2 Experiences in Software Testing

### 2.1 Start of My Testing Journey

3. Picture the scene. It's the 1990s, and it's the first day of my summer vacation job at a large telecommunications company. My manager hands me a huge book of test requirements and associated test cases and tells me that my next six weeks will be spent using them to test a PABX (Private Automated Branch Exchange) system and some VoIP (Voice Over IP) phones.
4. So that's what I did. I spent six weeks sitting in a lab for 12 hours a day, running thousands of manual tests. I wasn't a domain expert, so I wasn't sure what PABX or VoIP meant. I wasn't even an expert manual tester with a stack of testing patterns. I was just a rookie following each test step, expected result and writing out the actual test result.
5. At the end of the six weeks, I went back to my manager. I told him I'd completed all the tests, and none had failed. I assumed he'd be happy his software had passed every single test. As you can guess what he said. "I know. We've run them every year for the last decade. The tests have never failed."
6. But because I didn't know any better, my heart sank. At the time, it felt like I'd just wasted my entire summer. I hadn't added any value because I'd only proved something that had always been true was still true. None of my work had revealed anything new. All I'd done was take a horrible chore off someone else.

### 2.2 Automation was the Future Once

7. Once my disappointment had faded, I determined that I would automate the process, so it never had to be done manually again. I spent the next three years fully automating the tests so they could be run more frequently and more quickly. Most importantly, they would never have to be done manually again.
8. Until very recently, automated testing has been our best possible solution to maximizing test coverage, freeing testers to focus on work that adds more value. What's changed in the past year or so, of course, is the advent of Generative AI and Large Language Models (LLMs).

### 2.3 Generative AI is the Future Now – But Not as We Know It

9. I'm guessing that, like most people, you've spent the past year or so exploring what gen AI has to offer. You might be using it to write code. You might be using it to write essays or articles. You might be using it like you used to use Google to get answers to questions.
10. It's an extraordinary tool that clearly has huge potential for testers – but in its current form it has huge limitations.
11. It has no domain-specific knowledge and doesn't know any of the answers. For now, it won't tell you where it's got information from so, without your own domain-specific knowledge, you have no way to assess the validity (or otherwise) of the response. This will undoubtedly change in the next year as the EU AI Act comes into force, forcing LLMs to reveal their sources and give data owners the power to restrict the use of their data.
12. It doesn't offer any data privacy either. Doubtless we've all seen the stories of software engineers feeding proprietary code into ChatGPT and then finding the code has been used as training data for the system. The reality is that general tools like ChatGPT don't have much (if any) practical application in our world right now. It doesn't know what you know so it's in effect a generalist

advising a specialist. You can't use it as a co-pilot because you can't – or shouldn't – feed it any proprietary data or mission-critical information.

## 2.4 Use Your Proprietary Knowledge with No Risk

13. The offline nature of Generative AI is super-important because it means you can feed it your IP-sensitive data. You can give it your confluences, specification documents, and entire test repository, safe in the knowledge it will all stay within your own environment.
14. It gives you a tool that understands testing, your industry vertical, and your specific environment. It can generate fully automated tests and optimize and prioritize test execution through a digital twin model-based testing approach.

### 3 Unlocking Superpowers for Testers

#### 3.1 Value Driven Testing

15. At the heart of this paper is my simple concern that testing is too often seen as a cost to the business, not an enabler that adds value. I'll explore why that's the case – and show why and how gen AI holds the answer to improving the QA function's reputation.
16. But let's start by considering how much the IT landscape has shifted because I think that holds the key to understanding where it's all gone wrong.

#### 3.2 From IT as King to User as King

17. It's a truism because it's true: the world of IT regularly changes out of all recognition. At the start of my career, the web was just getting going, and IT in business largely consisted of mainframe applications, client service, and thick client systems.
18. It was a completely different landscape in another sense too. Users were expected to follow the workflow of the application not vice versa. If you didn't follow the workflow and something didn't work, it was your fault, not the system's fault.
19. Testing built up around similarly rigid architectures, with exacting governance, standards, and organizations around them.
20. Fast forward to today, and we have multiple devices, multiple applications, and multiple connectivities. The importance of user experience (UX) means applications must work how users want them to, not vice versa.
21. It's all made testing a thousand times more complicated than it used to be. Yet in the same time, testing hasn't really evolved. We still have relatively rigid testing architectures, governance, and standards.
22. And it gets worse too.

#### 3.3 Choose Two of Better, Faster, Cheaper

23. Competition is fiercer than ever. Whether it's your internal IT systems or your customer-facing systems, your organization needs faster release cadences to enhance customer satisfaction and drive competitive advantage. It also needs you to deliver at lower cost with better quality.
24. Yet, as we all know, when it comes to the well-known product triangle of better, faster, cheaper, you can only choose two.
25. Agile and DevOps helped us to speed up release cadences and introduced the concept of the minimum viable product.
26. But testing struggles to keep up with DevOps speeds, and that's a problem that isn't easily solved.
27. You can increase speed by lowering testing requirements, but this reduces quality. You can increase quality by maintaining testing requirements, but this reduces speed. You can throw resource at testing to increase speed and maintain quality, but this increases cost. In the first two

scenarios, you damage the user experience. In the third, you cost the business more than it wants.

- 28. Ultimately, the result is that the business starts to lose faith in the IT function's ability to deliver what's required – solutions that are better, faster, and cheaper. At the heart of this reputational issue is whether the business views the QA function as a cost or an enabler.
- 29. Too often, it's seen as a cost. Aside from the difficulty in delivering on better, faster, and cheaper, there's an even more fundamental presentational problem. Put simply, you spend time and money running a thousand tests. You find 20 defects – which will need time and money to fix.

### **3.4 Testing is a Critical Business Enabler**

- 30. In fact, QA should be viewed as an enabler. We're doing what the business needs us to do – driving the quality that will enhance the UX, increase customer satisfaction, boost reputation, and deliver competitive advantage.
- 31. The question is, then, how do we deliver testing better, faster, and cheaper so we can restore testing's reputation and have it viewed as a critical business enabler?
- 32. The answer is that you use Generative AI. Which turbo-charges the human-in-the-loop so the QA function becomes the value-adding team that gives the business what it wants.

### **3.5 Supercharge the Human-in-the-Loop**

- 33. The critical component of a truly successful DevOps approach is someone who is a specialist in the business side of things and a specialist in the technical requirements of implementation and testing. There aren't many of those around! Most people are either a specialist in the business or a specialist in the technical side of things.
- 34. Generative AI is effectively a specialist in both. It has business knowledge and it has technical testing knowledge. When you layer humans-in-the-loop on top, you add in the knowledge worker with wisdom to be able to validate and verify that everything expressed by every stakeholder and every permutation is correct. It gives you the ability to translate the subject matter of the business into something the IT and the technical teams can use to build an application and automate its testing.
- 35. It speeds up the pipeline and helps people on both sides of the equation to collaborate and work together effectively.
- 36. In short, it means we can properly shift left, bring testing right into the requirements phase of development, embed UX from the start – and elevate the value of testing. It's a game-changer that helps us deliver what the business wants and needs.

## 4 Turning Testers into Superheroes

### 4.1 Gen AI Can Do the Heavy Lifting of Test Design

The real value of testers is our ability to think critically and design the tests that get to the heart of the software and what it needs to do. Of course, this test design process is where the heavy lifting really happens. It's also one of the areas where Gen AI can add the most value.

Let's take this example to understand how.

Imagine you need to test braking specifications for a car. The specifications are just a tiny segment of the tests that need to be run for the vehicle and its overall function. At the same time, the specifications are also safety-critical, so it isn't an area you can skim and hope for the best.

Gen AI has the capability and the capacity to do all your thinking in a fraction of the time. It can develop a model with all the required axioms and delve into a level and combination of incredibly powerful detail. It can look at every scenario for heat, ice, rain, and more, individually and in combination. It can factor in road material and assess how tolerances will change.

This ability to interrogate every nuance has the potential to save incredible amounts of time in test design. Its comprehensiveness brings another benefit too, which I'll look at next.

### 4.2 Gen AI's Lack of Wisdom Can Be a Benefit

You can think of Gen AI as knowing everything and knowing nothing. That's because it lacks the wisdom that we bring to a problem as human testers.

To take the braking specification scenario again, Gen AI will give you every single possible permutation. Of those, some will be physically impossible or simply not reflective of the real world. Your wisdom brings the necessary X factor to the process – you'll be able to see which tests don't make sense or won't be needed.

At the same time, Gen AI's comprehensiveness can add value. As humans, we all have cognitive biases, which can lead to problems in software testing.

There's a story, probably an urban myth, about the F-16 Fighting Falcon fighter aircraft. The story goes that it flew perfectly well in the northern hemisphere – but it flipped 180 degrees the moment it crossed the equator. Why? Because cognitive biases meant the northern hemisphere developers and testers hadn't considered the southern hemisphere in their thinking.

It's an extreme example, but it's the sort of thing that happens all the time. Software designed for the 5G and full-fiber connectivity that every city enjoys but which hasn't yet reached rural communities. Technology that works in flat Illinois but can't cope with mountainous Colorado.

In each of these cases – and countless others – the developers' and testers' cognitive biases created issues. Gen AI helps eliminate those cognitive biases because it lacks the wisdom to have them in the first place.

### 4.3 AI Won't (And Can't) Take Your Job

The popular media perception is that gen AI is coming for all our jobs. Let's be clear: it can't, and it won't. Gen AI and LLMs draw on vast amounts of information and data – more information and data than a human could ever digest (or remember). What they lack is the next level – the wisdom and experience that humans bring to that information and data.

I liken it to Neo and Morpheus in The Matrix. Neo knows everything there is to know about kung fu, but he's never done it in the real world. It's why Morpheus, who has no knowledge but real-world wisdom and experience, can beat him so easily.

It means AIs can do a lot of simple tasks incredibly powerfully and therefore turbo-charge your productivity. But fundamentally, the human-in-the-loop will always be needed because they bring the wisdom that's needed.

But let's not underestimate the power of gen AI to do the simple things well. This leads us to the next point...

#### **4.4 LLMs Help You Achieve Work-Life Balance**

Think for a moment about how brilliantly collaborative the world of testing is. If you're struggling with a line of code or don't know the answer to something, you just ask. You might ask Google and trawl through the results. You might ask a colleague on Slack. You might ask a question in a user community.

Whichever way you choose, 99% of the time you'll get the answer you need. The trouble is, it takes time. You've had to wait to get the answer you need, so your task has taken longer than you'd like. Your colleague or the people in the user community who answered your question took time out of their days to help you.

You can think of LLMs as the latest iterations of this way of working. You ask them a question and they give you an answer – instantaneously. It gives us the potential to achieve a level of work-life balance that's eluded us in the past.

But to achieve that work-life balance, you have to know what you're doing – something we'll cover in the next point

#### **4.5 Getting the Best from Gen AI Depends on How You Talk to It**

As we just explored, gen AIs lack your context-specific wisdom. If you want to get the most from them, you need to know how to talk to them to give them that context. This is where prompt tuning comes in.

Firstly, you'll need to give them the persona they need – by telling them you're a Gherkin engineer or an MIT professor, they know the level they need to be operating on. You'll likely need to refine your questions – or prompt tune – to get to the real answer you need. At the same time, you can relax knowing you can ask it a stupid question without fear of being judged.

What's interesting, though, is how open LLMs are evolving – and getting lazier. Where earlier iterations would give you the entire code base you needed, today's iterations may only give you a snippet. To get the full code base, you need to explicitly ask for it, even to the point of saying, "I'm feeling lazy today, I don't want to write any of the code and want you to do it for me."

#### **4.6 Think of Gen AI as the Specialist That Complements You as a Generalist (And Vice Versa)**

Throughout software development history, the pendulum has swung from specialist to generalist and back again. One minute, everyone wants generalist full stack developers. Then next you need to be a specialist security engineer, a software tester, or a front-end developer.

#### **4.7 Gen AI and LLMs can help you be better at being both.**

When first started utilizing Generative AI, we fed it a wealth of testing-specific information. We then needed to help it understand relationships and dependencies so we could use RAG (retrieval augmented generation) architectures to enable it to deliver users the right information at the right time.

This means Generative AI is a generalist in that it knows – and can access – everything testing-related. When you're new to a test field, it's the specialist you need to get you up to speed. What's more, it's quicker than Googling and less embarrassing than asking a colleague.

It also means that when you're a specialist in a field it can be the generalist you need when something crops up that's outside your knowledge domain.

This brings us back to the collaborative nature of our world – and my final point...

#### **4.8 A Test Pilot in a Post-Pandemic World**

Pre-2020, we were all sat together in offices. We could celebrate the wins and commiserate the fails with our colleagues around us.

These days, we're likely remote workers. Who do you celebrate and commiserate with? Answer: your gen AI buddy. It's endlessly patient when you're working through the problem together, and it'll celebrate with you when you get something to work. Your gen AI stops being the monster that's going to take your job and starts being the selfless colleague who'll help get you your promotion.

Because, to come full circle: an AI won't take your job, but a human using an AI will.

## 5 Conclusion

This paper provides a reflective and forward-looking analysis of the evolution of software testing, underscoring the journey from manual processes to the integration of advanced technologies like Generative AI. This paper has delved into personal anecdotes and broader industry trends to illustrate the shifts and transformations that have shaped the current landscape of software testing.

The introduction of Generative AI and Large Language Models (LLMs) marks an even more transformative era in software testing. While automation addressed the inefficiencies of manual testing, Generative AI offers unprecedented potential to enhance and revolutionize the testing landscape. Despite its current limitations, such as the lack of domain-specific knowledge and concerns over data privacy, Generative AI has demonstrated its capacity to handle vast amounts of data, generate comprehensive test scenarios, and reduce cognitive biases. Its ability to operate offline, using proprietary knowledge securely, presents a significant advantage in maintaining data integrity and confidentiality.

The core argument of this paper is that testing should be viewed not merely as a business cost but as a critical enabler of quality, user experience, and competitive advantage. The evolution from rigid, workflow-centric applications to user-centric designs has significantly increased the complexity of testing. Organizations today demand faster release cadences, superior quality, and cost-efficiency — a trio that traditional testing methods struggle to balance. Integrating Generative AI with human augmentation offers a solution to this conundrum, enabling more efficient, comprehensive, and nuanced testing processes.

Generative AI can significantly reduce the heavy lifting involved in test design and execution. By generating exhaustive test scenarios and permutations, it ensures comprehensive coverage and helps eliminate human cognitive biases. However, the irreplaceable value of human testers lies in their wisdom and contextual understanding, which are crucial for validating and verifying AI-generated outputs. This human-in-the-loop approach ensures that while AI handles the data-intensive aspects of testing, human expertise provides the necessary oversight and nuanced judgment.

The future of software testing lies in this symbiotic relationship between AI and human testers. AI will not replace human testers but will augment their capabilities, making them more efficient and effective. This collaboration will enable testers to focus on strategic, high-value activities, ultimately enhancing the reputation of the quality engineering function as a critical business enabler.

In conclusion, this paper envisions a future where the integration of Generative AI transforms the testing landscape. By leveraging AI's data processing power and human wisdom, organizations can achieve better, faster, and cheaper testing outcomes. This evolution will not only restore the reputation of the quality engineering function but also elevate it to a strategic role within the business, driving quality, innovation, and competitive advantage. The journey from manual testing to AI-enhanced processes underscores the dynamic and ever-evolving nature of software testing, promising a future where technology and human ingenuity work hand in hand to deliver exceptional software solutions.

## 37. References

Adam Leon Smith (Edited by), Rex Black (author), James Davenport (author), Joanna Olszewska (author), Jeremias Rößler (author), Adam Leon Smith (author), Jonathon Wright (author). 2022. *Artificial Intelligence and Software Testing: Building systems you can trust*. British Computer Society

Dorothy Graham (author) / Mark Fewster Graham (author), Jonathon Wright (author), Experiences of Test Automation: Case Studies of Software Test Automation, 2012, Addison-Wesley Professional

Eran Kinsbruner (Edited by), Jonathon Wright (author), The Digital Quality Handbook, 2006. Infinity P

# Data Quality Assurance in Credit Card Fraud Prevention

**Yuqing Yao**

yqelisa@gmail.com

## Abstract

In 2023, credit card fraud cases soared to 425,977, up 53% from 2019 driven by the surge in online shopping amid the COVID-19 pandemic. This underscores the pressing need for advancing Fraud Prevention techniques, heavily relying on data analysis and modeling.

While modeling techniques for fraud detection are widely discussed in academia, data quality assurance for it is a less popular topic. However, with the challenge of data imbalance and the manual components in fraud labeling, data quality assurance becomes even more important to ensure the quality of fraud detection. This article discusses types, causes, and solutions for data quality assurance in fraud detection, addressing:

1. Impact of data quality on credit card fraud detection
2. Challenges specific to fraud detection data quality assurance
3. Types of datasets used in this field
4. Tools and procedures
5. Causes of data quality issues based on different dimensions
6. Solutions for data quality issues

Besides reviewing recent academic research papers, this article incorporates industry research reports, technical blogs from leading FinTech companies, and vendor product specifications. It focuses on actionable insights for data quality assurance in credit card fraud detection.

## Biography

*Yuqing Yao currently works as a Senior Fraud Analyst at a FinTech company, Klarna. She uses statistical approaches and visualization to detect fraud patterns and program them in the fraud decisioning system. Prior to that, she worked as Assistant Director in Data Operations at Moody's Analytics, leading the data processing and reporting of its credit data consortium. She started her career in the Treasury team at Stripe, doing liquidity forecasting and monitoring. Her career interests have been in data science, analytics, and risk management. In her spare time, she actively volunteers for organizing AI/ML meetups in San Francisco. Besides, she is also a long-term Toastmasters.*

Copyright Yuqing Yao 2024-06-23

# 1 Introduction

## 1.1 Why should we care about Credit Card Fraud Prevention

Credit cards brought us convenience and enabled us to use leverage, thus becoming an essential part of the economy. As of 2023, credit card is the top payment method by volume and the second largest payment method for online transactions by volume (Worldpay 2024).

Fraud prevention protects customers from bad actors. Based on the U.S. Federal Trade Commission (FTC) report, consumers incurred over \$10 billion fraud losses in 2023 (Federal Trade Commission 2024), which is approximately 150,000 US households' annual expenditure in that year. We can't stop fraudsters from having ill intents, but we can protect the financial institutions and consumers from those bad actions. Therefore, keeping fraud prevention methods and technologies up to date is how we win this cat-and-mouse game.

## 1.2 The role of data in Credit Card Fraud Prevention

Fraud prevention heavily relies on data to differentiate fraudsters from genuine customers. Data is used in multiple ways in the lifecycle of fraud prevention:

- **Behavioral analysis:** Use customer activities to find loopholes in product design that let fraudsters in.
- **Detect fraud attacks:** Detect abnormal volumes and trigger alerts.
- **Case studies:** Extract patterns from fraudsters' behaviors to understand how they conduct frauds.
- **Real-time decisioning:** Use data to inform which customer or transaction should be blocked.

## 1.3 Why is data quality important in Credit Card Fraud Prevention

### 1.3.1 Data Driven Decisioning

According to a survey in the Insurance Fraud space (SAS 2021), 64% of the respondents think that poor data quality is the most significant challenge for implementing fraud detection technology. Similarly, fraud prevention for credit cards requires a high level of automation, because of the large volume of sign ups and transactions. How we tell the fraudsters or fraudulent transactions apart from the legitimate customers and transactions is largely dependent on recognizing the patterns of these two groups. This means that the quality of the data we collect affects the quality of our decisions.

### 1.3.2 Class Imbalance

Identifying frauds from non-frauds is a classification problem. Fraudsters are usually a very small percentage of the sign up or transaction attempts making the sample size of this class much smaller than the other class, which creates a big challenge for classification models. Despite having ways to handle the imbalance, upsampling for example, the lack of data still makes training sensitive to small data errors in the fraud class.

### 1.3.3 Fraud as Anomaly

Anomaly Detection is also widely used in fraud detection. By definition, anomalies usually have extreme values or weird patterns compared to normal observations. If a normal customer's data has errors while being collected, the customer could be wrongly categorized as a fraudster, even though the actual values are normal. This increases False Positives and creates negative customer experience.

## 1.4 Scope of this Paper

This paper reviews academic papers about data quality and fraud analytics, to find the intersection of these two topics, which is not widely explored but practical. It sorts the problems and solutions for data quality in credit card fraud prevention into a logical framework, making them easy to comprehend. It also synthesizes the latest best practices from technical blogs of leading global FinTech companies and industry conferences to map out different solutions.

The rest of the paper is organized as follows: Section 2 lays out the types of datasets used in Credit Card Fraud Prevention to show the objects of quality assurance. Section 3 lists the tools (technologies and processes) used in this area. Section 4 discusses the commonly seen data quality issues and the cause of them. Section 5 proposes solutions to the different data quality issues. Section 6 summarizes the work and suggests future improvements.

## 2 Types of Datasets

Fraud prevention uses different types of data to paint a holistic picture of the customer. Each type of data has some unique data quality problems.

### 2.1 Customers

The first step of blocking credit card fraud is blocking fraudsters from signing up. This involves collecting customer information and verifying if it matches the real customers. Typical fields being collected are:

- First Name and Last Name
- Date of Birth
- National Identity Number
- Billing / Mailing Address
- Occupation
- Email
- Phone Number

Fraudsters could steal a real customer's identity and sign-up credit cards on their behalf, leaving the real customer liable for the credit card debt, or the financial institution to eat the cost. This type of fraud is called Identity Theft.

Alternatively, fraudsters could make up fake identities and try to fool the financial institution to issue credit cards. For example, making up a fake name, but matching with corresponding identity documents to make the sign-up material seem legitimate. This is called Synthetic Identity.

To block these types of fraud, we compare the self-reported information during the sign-up process with the data previously on file, or with external bureaus, to see if they are consistent.

### 2.2 Transactions

If the customer's account is hacked, or the customer's card information (card number and CVV) is stolen, fraudsters could use a card from a legitimate customer to make purchases. These frauds are called Account Takeover and Card Skimming, correspondingly.

Sometimes a customer does not want to pay back the credit card debt, the customer claims someone else (a fraudster) made those transactions, hoping to shift the loss to the financial institution, this is called First Party Fraud.

In both situations, data of the specific transactions is critical in identifying which transactions are fraudulent and which ones are not. The following data points are interesting to fraud prevention:

- Card Number
- Card Verification Value (CVV)
- Billing Address
- Expiration Date
- Transaction Amount
- Transaction Currency
- Merchant Name
- Transaction Description
- Entry Mode

## 2.3 Behaviors

Besides looking at the customer information at sign up and individual transaction information, we also look into customer behaviors over time.

For example:

- Log in Device
- Log in Time
- Log in Location
- Purchased Items
- Average Purchase Amount

When we observe a different pattern compared to this customer's previous behavior, we would suspect it's a fraudulent transaction or log-in.

## 2.4 External Data Sources

There are external data providers that supply data points not directly collected through the financial institution's interactions with the customer. For example:

- Credit bureau data
- Background check data
- Social media data
- Fraudulent reports from other financial institutions

## 2.5 Datasets Summary

Each aforementioned set of data alone is not enough for identifying fraud. The more comprehensive the datasets are, the fuller picture we have of the sign up or transactions. Used in isolation, each of them has their own set of problems. However, them being used together creates a new set of problems, which will be explained in section 5.2.

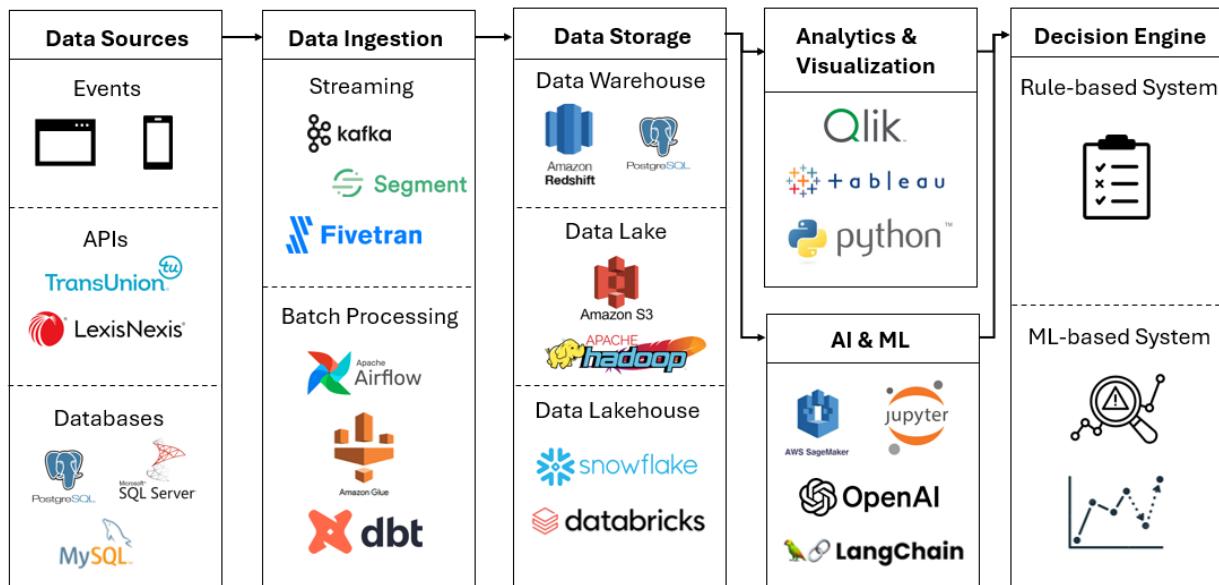
# 3 Data Stack

A Data Stack is a collection of technologies and processes that enable data to be collected, processed, and then used. Fraud prevention shares a similar data stack of many other e-commerce analytical problems. Understanding the data stack helps us identify where the data quality problems are introduced.

## 3.1 Data Stack Diagram

The below diagram shows how each of the components interact with each other, and where they are positioned in the workflow for fraud prevention.

- Data enters the system from sources through ingestion tools.
- Then the data is pushed to be stored in some structure, Data Warehouse, Data Lake, or Lakehouse.
- From there, we can consume the data by visualizing or generating predictions.
- Finally, we use the insights from analytics or modeling to trigger actions in the Decision Engine, where sign-up or transaction attempts are marked with “accept” or “reject”.



## 3.2 Data Sources

The previous section, we looked at the types of data used in Fraud Prevention. Some types are collected directly from interacting with the customer, and some come from external parties:

- **Customers:** customer submission through online forms
- **Transactions:** collected from the card network
- **Behaviors:** customer frontend events
- **MERCHANTS:** collected from the card network
- **External Data Sources:** integration with data vendors and credit bureaus

## 3.3 Data Ingestion

Data ingestion is the process of obtaining, importing, and processing data from multiple sources into a single storage place. The purpose of data ingestion is to standardize and centralize data for different teams in the organization to use. Data storage locations will be explained in the next section.

Depending on whether data is processed as it is generated, or first being loaded then being processed altogether, we have batch processing or streaming. Common choices among these two are:

- **Streaming:** real-time fraud decisioning data (transactions), front-end events
- **Batch processing:** non-time-sensitive fraud variables (statistics over a certain time window, segment benchmarks), analytics, and reporting

### 3.4 Data Storage

Based on the needs of the organization and nature of data, there are following forms of storage (Striim 2024):

- **Data Warehouse:** a unified data repository for storing a large amount of data, usually in structured format.
- **Data Lake:** a centralized and flexible data repository that stores both structured and unstructured data in its raw form.
- **Data Lakehouse:** A hybrid of warehouse and lake, which enables storage of both structured, semi-structured, and unstructured data. It's built upon data lakes that stores all the raw datasets, and then applied ACID transactional processes that are data warehouses' features.

Because fraud prevention involves a fair amount of both analytics and Machine Learning, data lakehouse is gaining popularity in major FinTech companies. Thus, credit card fraud detection is also often conducted based on the lakehouse architecture.

### 3.5 Data Analytics

Analytics include data cleaning, transformation, and calculation. This is when insights are drawn from data, and then guide actions.

Fraud detection involves a lot of deep dives and ad hoc analysis. These types of analysis are usually done in Jupyter Notebooks or spreadsheets. Lakehouses usually support SQL querying, and that is how analytics is done in many cases.

In some cases, vendor data is only accessible through the vendor environment, which makes these vendor software applications a part of the analytics component of the data stack.

### 3.6 Machine Learning and Artificial Intelligence

More advanced fraud prevention systems use Machine Learning to support decision-making. This usually includes functions for:

- Feature Engineering
- Model Training and Evaluation
- Model Deployment
- Continuous Monitoring and Updating

### 3.7 Visualization and Reporting

Visualization serves multiple purposes for fraud prevention:

- Build intuition on fraud trends and patterns
- Detect anomalies and surface potential fraud attacks
- Showcase fraud prevention engine and algorithm performance to stakeholders

The tools for visualization include:

- Dashboards
- Jupyter Notebooks
- Reports (usually in PDF format)

### 3.8 Decision Engine

This is where insights are turned into actions. It's the system that streams transactions or card opening applications data, go through a set of rules defined by the card issuer to calculate the fraud risk, and come up with either accept or reject decision based on that.

The following categories of decision engines are common in credit card fraud prevention:

- **Rule-based systems:** Analysts discover fraud patterns using analytics, and then define a set of rules each transaction or sign-up needs to go through. This is usually easy to comprehend, because the thresholds and conditions used in these rules have context, for example, a fraud attack, or a data breach.
- **Anomaly detection:** It's also relatively easy to comprehend, because it flags frauds by how different the values are from normal transactions.
- **Predictive Analytics:** Use statistical models or machine learning models to generate a score for transactions or sign-up attempts. Depending on the type of model used, the interpretability varies.

## 4 Data Quality Issues and Causes

### 4.1 By the Nature of Data

#### 4.1.1 Imbalanced Data

When we build classification models to predict whether a customer or transaction is fraudulent or not, having very few observations of frauds versus the majority are non-frauds creates the class imbalance problem. In the real world, fraudulent transactions and signups are the minority. According to research from the Federal Reserve Bank of Kansas City (Hayashi 2019), credit card fraud rate in the US for no chip credit cards is about 16 basis points (0.16%) in 2016, and that of chip credit cards is about 12 basis points (0.12%).

This is because people who want to commit frauds are the minority of the population. Based on the theory of "Fraud Triangle", someone needs to have the motivation, rationalization, and opportunity to commit frauds (National Whistleblower Center 2023). Besides, financial institutions actively employ fraud prevention methods, so even though there could be fraud attempts, the successful ones are quite few. When we need a clear label for fraud, we only consider those attempts that got past the prevention methods and eventually confirmed to be fraud either by disputes or chargebacks.

Having imbalanced data could cause the model training to be biased towards predicting the majority class. For example, if only 1% of the samples are real frauds, then if a model predicts 100% of the samples to be frauds, it still has 99% accuracy. Besides, even with methods like oversampling, it could cause overfitting to the limited number of fraud samples.

#### 4.1.2 Wrong Labels

As the target variable, fraud or non-fraud as a binary value is usually sourced from chargebacks or customer disputes. Even though there are certain patterns to infer fraud vs non-fraud, sometimes drawing conclusions for the cases still rely on human investigation. For example, when a customer disputed the transaction as "unauthorized purchase", it could be that the customer is telling the truth, and it's a third-party fraud, but it could also be that the customer lied about it, and in this case it's a first-party fraud. The patterns of these two types of frauds are different and mis-labelling the records leads to worse model performance.

Besides, mis-labeling could also come from manual entry errors. For example, if the agent handling the dispute closed the case with reason "wrong merchandise" instead of "unauthorized purchase", then it's considered non-fraud in subsequent analyses, while it actually should be fraud.

### **4.1.3 Incomplete Data**

Credit card signup data has a lot of personal information, so it's subject to strict regulations in most regions such as the General Data Protection Regulation (GDPR) in the European Union. In the case of GDPR, consumers have the right to opt out of certain data collection, which could leave fewer attributes for companies to use for analytics, including fraud detection. Individuals also have the right to have their data deleted, limiting the types of data available.

## **4.2 By Procedures**

### **4.2.1 Latency**

During the data ingestion phase, there could be large backlogs of tasks to be run, creating lag between data being generated and data being loaded into storage. This lag could create a gap between when a fraud attack happens and when teams are able to get the relevant data to conduct root cause analysis. Based on industry case studies, this gap could be 1 hour (Materialize 2024) up to several days. Even just 1 hour of lag could cost companies thousands to tens of thousands of dollars.

### **4.2.2 Wrong Calculations**

Transformation processing is using filters, operations, and functions to turn the source data into desired format and metrics. When Analysts or Data Engineers write the transformation logic, they could make mistakes, resulting in wrong calculations. For example, when joining different tables together to form new tables, missing a join key could create duplicate records. If metrics are calculated based on duplicated records, it could deviate from the actual metrics.

### **4.2.3 Data Silos**

Credit card issuers collect data from different sources, and the integration and ingestion processes could be owned by different teams. Without a proper Data Catalog, teams don't always know what each other is processing. This creates data silos, which means different sets of data exist, but are not being used together.

For example, the Marketing team could send out a large campaign, causing credit card sign-up applications to significantly increase in a short period of time. If the fraud team is not informed of this campaign, they would have a hard time identifying whether this application volume spike is from a fraud attack or because of a marketing campaign, which mainly brings in non-fraud customers. This causes hours to days of labor that could have been saved if there's a clearer visibility of datasets.

## **4.3 By Nature of the Task**

### **4.3.1 Afterthought of the Business**

Because data quality assurance is usually considered an operational or back-office function, it does not get as many resources as revenue-generating functions, such as Product or Sales. When data quality is high, it does not get noticed because things are working as expected. However, when the data quality drops and causes incidents, that's when it gets visibility.

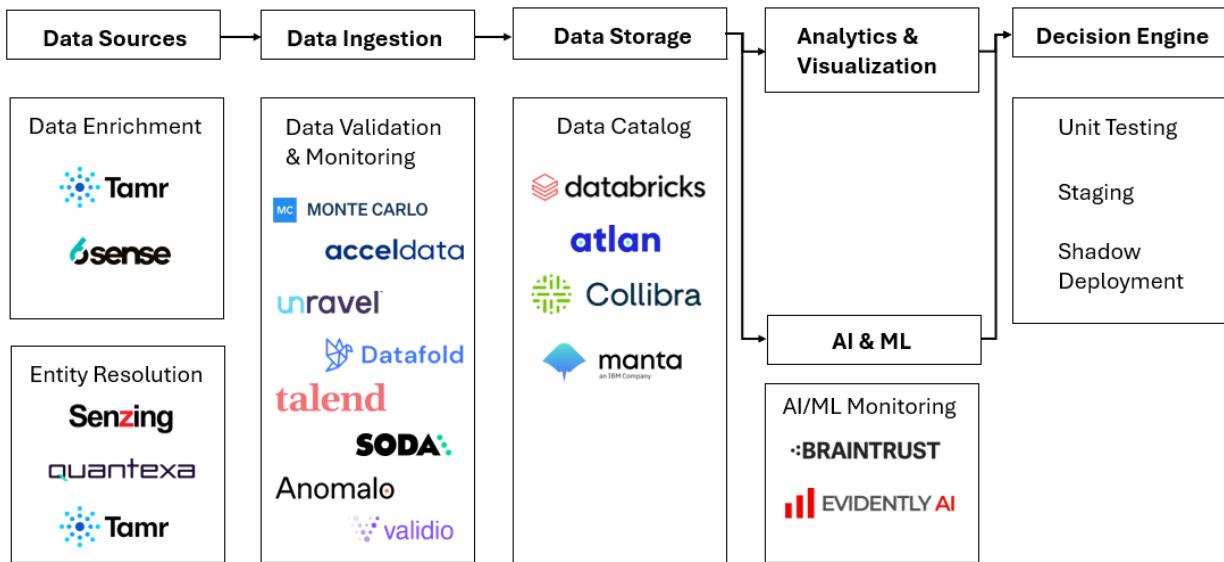
### **4.3.2 Data Quality Assurance could be Tedious**

Because data quality assurance involves a lot of ad hoc analysis to find out the root cause, it could be tedious. It also does not have very tangible output, so it doesn't usually provide much sense of fulfillment. Therefore, it's hard to attract talents to work in this field, even though it's an important function. When there's a shortage of talents to work on it, there aren't sophisticated enough checks on data quality.

## 5 Data Quality Assurance Solutions

### 5.1 Solutions Diagram

This diagram maps the solutions and representing products / companies to the Fraud Prevention data stack.



### 5.2 Data Validation

Data validation is the process of validating that the data has the expected data type, range, and format etc. In Fraud Prevention, common validation checks include:

- **Data type:** Customer names should be in string type, and date of birth should be in date type.
- **Range:** Transaction date should be after the card was created; date of birth doesn't exceed biological constraints (e.g. not possible to have a customer who is 200 years old)
- **List of values:** Dispute reasons need to conform industry standards, for example, "unauthorized purchase", "wrong merchandise".
- **Format:** National Identification Number should have the same format for the same country.

Data validation could be built into the user interface for data entry. For example, when customers enter their street address, have them select from USPS verified addresses. Internally, the company can build standardized labels for fraud investigators to choose from when they close the cases, instead of having free text entry as labels. Besides, there should be sampling on the labelled cases, to verify that the agents didn't tag the cases randomly to sacrifice the quality for speed.

### 5.3 Entity Resolution

Entity Resolution matches different records that are referring to the same thing and resolve them into the same entity. For example, a customer could have a legal name and preferred name. Using entity resolution methods, we can link these two different objects together and get a more holistic view than without it. This is especially helpful for fraud linking analysis, which leverages graph structure, because it splits the same entity into different nodes when the source records have different attributes.

Therefore, introducing Entity Resolution to fraud analytics improves data quality and creates positive business impact. According to Entity Resolution provider Quantexa, its entity resolution applied in fraud Excerpt from PNSQC Proceedings

prevention achieved an 80% reduction in investigation time and a 75% reduction of false positives (Quantexa 2024).

## 5.4 Data Enrichment

Data Enrichment is the process of integrating external data sources and combining it with internal data, to add missing details of the entity. For example, customers may only submit very few essential fields, but credit card issuers can query vendors or bureaus to get more information about the customer. This alleviates the problem of missing data.

## 5.5 Rule-based Monitoring

A key element in data quality assurance is establishing metrics and monitoring them. There are some commonly used metrics across different domains like percentage of nulls in a column, percentage of anomalies in a column, number of duplicated records. Business logic determines what metrics should be used, for example, what is considered an anomaly – shall we use 2 standard deviations away or 3 standard deviations as the threshold. Rule-based checks could be done at each destination of data loading. However, because this approach is not very scalable, many organizations only use it for critical datasets. Discover observes that it takes about 40 hours for a Data Analyst to design a quality check for a column (Jaganathan 2023).

## 5.6 Data Catalog

Data Catalog is a metadata repository of datasets in the organization which includes but not limited to Data Dictionary, Data Lineage, and Data Ownership. It is an important tool to manage datasets and improve data quality.

- A **data dictionary** has field names and explanation of the dataset, which helps users understand how to use it. This helps identify misusage of fields and helps define validation rules.
- **Data lineage** shows the upstream and downstream datasets, which helps assess impact of data issues for downstream processes, and trace back the root cause through the upstream processes.
- Showing **data ownership** helps quickly resolve data issues when they are discovered.

## 5.7 ML-Powered Monitoring

Machine Learning (ML) techniques automatically discover patterns and train for the best parameters for predictions. Using ML to power data quality monitoring frees Data Analysts from manually defining rules. For example, using Unsupervised Learning techniques to conduct data quality checks helps discover problems humans cannot think of. For example, a human analyst could define intervals to use for alerting anomalies, but it's hard to detect seasonality in data.

Typical models used in data quality monitoring include:

- **Tree based models:** Easy for interpretation and implementation; useful for detecting categorical inconsistencies.
- **Support Vector Machines (SVM):** Can be used for separating high quality and lower quality data points; scalable.
- **K-Nearest Neighbors (K-NN):** Used for imputing missing values.
- **K-Means:** Group similar data points together and identify anomalies that don't belong to larger groups.

## 6 Conclusion and Future Improvements

This paper proposed solutions of data issues in Credit Card Fraud Prevention by looking at the object of the issues, processes, and tools involved.

Firstly, it highlighted why data quality is essential in the problem space. Then, it summarized the types of data used in this space: customer information, transactions, behavioral data, external data sources like bureau data and social media data. The diversity of data sources contributes to the complexity of data management and problem of data silos. Afterwards, it illustrated the commonly used data stack in Credit Card Fraud Prevention, using a diagram to show the relationships of data sources, ingestion pipelines, storage, analytics, visualization, ML/AI, and decisioning. It further discussed the causes of data issues through different dimensions, including the nature of data, procedures, and the nature of the tasks. Lastly, it suggested solutions including data validation, entity resolution, data enrichment, data monitoring, and data catalogs.

For future work, we will use real-world case studies of FinTech companies to show the impact of applying the best practices of data quality assurance, and how that prevents losses. Besides, as Artificial Intelligence possesses the center of the stage in technology, we would discuss how AI is used in Credit Card Fraud Prevention and how AI observability should be conducted in this field.

## References

- Atlan. n.d. "6 Reasons Why Data Quality Needs a Data Catalog." *Atlan*. <https://atlan.com/data-quality-with-data-catalog/#:~:text=A%20good%20data%20catalog%20can,can%20inform%20data%20quality%20measures>.
- Federal Trade Commission. 2024. "As Nationwide Fraud Losses Top \$10 Billion in 2023, FTC Steps Up Efforts to Protect the Public." *Federal Trade Commission*. 2 9. <https://www.ftc.gov/news-events/news/press-releases/2024/02/nationwide-fraud-losses-top-10-billion-2023-ftc-steps-efforts-protect-public>.
- Hayashi, Fumiko. 2019. "Payment Card Fraud Rates in the United States Relative to Other Countries after Migrating to Chip Cards." *Economic Review*, 12. Accessed July 2024. [https://www.kansascityfed.org/Economic%20Review/documents/681/Payment\\_Card\\_Fraud\\_Rates\\_in\\_the\\_United\\_States\\_Relative\\_to\\_Other\\_Countries\\_since\\_Migrat.pdf](https://www.kansascityfed.org/Economic%20Review/documents/681/Payment_Card_Fraud_Rates_in_the_United_States_Relative_to_Other_Countries_since_Migrat.pdf).
- Jaganathan, Prakash. 2023. "Discover's Approach to Scaling Enterprise Data Quality Monitoring." *Anomalo*. 10 17. Accessed 2024. <https://www.anomalo.com/case-study/discover-s-approach-to-scaling-enterprise-data-quality-monitoring/>.
- Materialize. 2024. "Real-Time Fraud Detection: Analytical vs. Operational Data Warehouses." *Materialize*. 3 7. Accessed 7 9, 2024. <https://materialize.com/blog/fraud-detection-latency-accuracy/>.
- National Whistleblower Center. 2023. *The Fraud Triangle*. Accessed 2024. <https://www.whistleblowers.org/fraud-triangle/#:~:text=According%20to%20Albrecht%2C%20the%20fraud,being%20inconsistent%20with%20one's%20values.%E2%80%9D>.
- Quantexa. 2024. "Create context to counter the rise of fraud." *Quantexa*. Accessed 7 10, 2024. <https://www.quantexa.com/solutions/fraud/#challenge>.
- SAS. 2021. *State of Insurance Fraud Technology Study*. SAS.
- So, Kenn, and Ben Lorica. 2020. *Gradient Flow*. <https://gradientflow.com/data-quality-unpacked/>.
- Striim. 2024. "Data Warehouse vs. Data Lake vs. Data Lakehouse: An Overview of Three Cloud Data Storage Patterns." *Striim*. <https://www.striim.com/blog/data-warehouse-vs-data-lake-vs-data-lakehouse-an-overview/>.
- Worldpay. 2024. *The Global Payments Report*. Worldpay.