Automation - We're Doing it Wrong

Melissa Tondi

melissa.tondi@gmail.com

Abstract

There are many mis-conceptions and perceptions around automation as it pertains to software testing, so in this talk, we will discuss those and set the record straight on what automation is and, more importantly, what it is NOT.

The term automation as it pertains to software testing has been a driving force in defining the software testing industry. For many years, we've used it as a catch-all to determine whether a tester, testing team, or IT organization is successful. In this talk, we will discuss the five misconceptions that are pervasive within companies - including using a percentage or number of test cases to define success and specifying a title/role for those who automate and matrixing that role across teams versus embedding it within the project they are supporting. Melissa will discuss re-booting our current thinking of automation and show tactics to address the five misconceptions that have contributed to what many of us would consider one of the wedges that divide the testing industry. In addition, she will share practical and proven approaches you can take back with you to show immediate and valuable results.

Biography

Melissa Tondi has spent most of her career working within software testing teams. She is the Director of Quality Engineering at Guild Education and a Principal Consultant at Disrupt Testing, where she assists companies to continuously improve the pursuit of quality software—from design to delivery and everything in between. In her software test and quality engineering careers, Melissa has focused on building and organizing teams around three major tenets—efficiency, innovation, and culture – and uses the Greatest Common Denominator (GCD) approach for determining ways in which team members can assess, implement and report on day-to-day activities so the gap between need and value is as small as possible.

1 Introduction

In this talk, we introduce the four approaches that we introduced while building test automation practices that may have made sense then but may have gotten stale now. These “wrongs” serve as a reminder that innovation should be at the forefront of our industry and even best-laid plans and ideas that make it in to our QE playbook should be re-visited to ensure they are applicable, align with the industry, and are collaboratively discussed within our agile teams in order to support the highest efficiency and productivity on our quest to deliver software quicker and with high quality. Here are the top “wrongs” I’ve gathered in hopes that our community can continue to continuously improve and broaden our information sharing.

2 Wrong #1 Monolithic Suites of Tests

When we invest in on-boarding activities for new hires, we are making a statement that it’s important to provide consistent information at the onset of a team member’s career with the company. Far too often, we don’t invest that same thought process to the rest of the team or re-visit processes put in place months (or, sometimes, years) ago to ensure they are still valid and supportive of what actually takes place.

We implemented a playbook titled Definition of Done. The playbook allows us to level-set all the services our team provides and present those to our agile teams.

2.1 Definition of Done Playbook Sample Sections

- What QA Does and How
    - Test Planning
    - Test Management
    - Test Execution both scripted and un-scripted
    - Reporting
- QA’s Expectations from Development, Design, Product Management, and Scrum Master

3 Wrong #2 Automate Everything

Fundamentally, we all know that not all tests should or can be automated, but I’m still surprised when I hear teams are being measured on the number or percentage of tests automated versus weighing the overall value the automation is bringing to the team. A better way to approach this is to have an intuitive selection process to quickly determine when something should be considered for automation. Notice how I used the word "considered" instead of giving instruction to automate a test. When you are held to a meaningless metric like “everything” your creative license is essentially removed and how fun is it to be told what to do rather than doing what you know is more valuable?

3.1 The Automated of Automatable (AofA) Metric

Because we did away with the “traditional percentage of test cases that are automated” metric that tends to be an inaccurate measure of software quality, we focused deeper on a metric that showed value, not only to the QA team, but to the agile project team as well. The AofA is determined by a selection process that succinctly shows the percentage of automated valuable tests given the following sample criteria:

- Its severity to the business in terms of:
    - Revenue Loss
    - Security Vulnerability
    - Customer Loss

- Its importance to customer happiness
- Compliance

Given these, during refinement sessions, we indicate which user story and/or acceptance criteria meets any of the above criteria and consider it for automation. Once the initial sorting happens, we size the work according to the project team’s norms and plan the work accordingly.

For reporting purposes (either daily or recurring throughout the sprint), we can report on the number of items that met our criteria and could be automated versus what was actually automated. We increased our percentages of AofA to the mid 90th percentile to 100% in most cases.

Using the guidelines in section 5, we worked in serial order of priorities. For example, if any test in our Priority 1 suite was not green, we swarmed to fix and did not expand any other automation until all higher prioritized automation was green.

4 Wrong #3 A Siloed and/or Non-Embedded Team of Automation Engineers

Many times when I’ve provided SDLC assessments for companies, I observe their team dynamics and any agile ceremonies that are followed. When I chat with QA and their management, there is a pattern of exclusion during key meetings where Development may have received more information than what was represented in the user story or its acceptance criteria. This usually results in QA not being able to size the work correctly or to assume behavior on “light” acceptance criteria. This then causes more triage by Product for bugs reported, mis-interpretation of intended behavior by QA and general tension between QA and other team members because they were not privy to adhoc or informal conversations that have taken place without them.

One option we’ve used to counteract that is smaller working refinement sessions. We do this by setting outcomes for refinement and have Product provide a list of prioritized stories ready to be refined at least 48 hours before the refinement session. We provide expectations for a story ready to be refined and have any one with responsibility on the story attend and contribute by providing a high-level summary of how they will approach the work. If any of those items are not complete during refinement, the story is not refined, and, therefore, cannot and should not be planned and committed to for an upcoming sprint or for Kanban, work should not be started. By holding smaller working sessions with those who have tasks, this creates a much more efficient, collaborative session where contextual information is being shared and heard, and implicit information becomes explicit and used for higher quality activities. It also ensures that each person and their practice are equally collaborative and represented for true sizing and estimates.

5 Wrong #4 Automation Deliverables are Owned, Executed and Maintained by Automation Engineers

In addition to the above, we know the Development team does their own testing, but sometimes we don’t know what that is. By defining who does what and creating a playbook of each agile pillar’s consistent practices at the individual contributor level, it removes ambiguity and take the guesswork out of what actually happens when that card or ticket moves to the “ready for QA” column. We advocate a collaborative discussion and refinement of work across the agile team to ensure test activities performed by QA are not only not redundant, but that are highly valuable and done at the right time during that phase.

6 Right #1 Multiple Runs for Multiple Dones

A while ago, we changed the name and mission of QA in our organization. We determined that Quality Engineering and how we defined it most matched what we were currently doing, where we eventually wanted to do, and, perhaps most importantly, where we knew our value would be emphasized the most within the organization. When given the opportunity, sometimes it's good to disrupt with a name change when a company or its leadership has misperceptions of what QA's role is versus what it should be.

This quote "Influence the Building of the Software before the Software is Built" drives our daily activities. We do this by:

- Balancing technical acumen with user advocacy and ensure we emphasize both.
- Using context-driven techniques. Given the information we have, we determine if it's enough and if not, we find more by:
    - More collaboration within Development, Product, other QE teams, Customer Support, and Customers.
    - Reaching out to the community. We both consume from and contribute to the community whenever we can.

In our QE playbook, we highlight the following traits and characteristics of the "Engineer" in Quality Engineer.

6.1 Breakdown of Engineering Traits

- Define success, outcome and measurements. Using context-driven approaches to gather the right data and information. Usually, the easiest information to gather is the that which is explicit (user stories, requirements, acceptance criteria, etc.). QE adds value when we uncover valuable information that is implicit at first.
- Design a comprehensive strategy.
- Build the solution. Write the tests, write the charters/sessions for unscripted testing.
- Execute the solution.
- Measure the results and rove them!
- Report the outcomes throughout.

By re-visiting and addressing any or all of these "do nots" you'll be operating at higher efficiency and productivity and can continue to innovate and continuously improve.

7 Right #2 Know Your Landscape

- Consider Unit/Integration tests and take action by designing a strategy (documented or not) that includes all testing being done across the team. One of the most valuable traits QE has is our ability to gather information across practices. We should use that trait to also build a comprehensive strategy that not only incorporates the tests we create and eventually execute, but considers the tests and testing that are also executed across the ream
- Start with Smoke tests. After inventorying all the tests and testing happening across the team, it's a great time to also build in your test automation strategy. I recommend starting with smoke tests, but, first, let's make sure it's defined and understood across the team. I like to define Smoke tests

by following the ISTQB definition as closely as possible and also adding a bit more refinement to it, so for this paper, we will define Smoke tests as following the CRUD (Create, Read, Update Delete) actions. Those should be considered the most basic functions and should also resonate across the team.
- Next, tie it to Definition of Done/Shippable. Now that we have a more manageable starting point for test automation, we want to use our Agile methodology to apply it to “done”. In this case, I recommend that we don’t consider the ticket/work done without

8 Right #3 Using the “Tests as a Service” Approach

Organize tests in the same way your software engineers/developers are. This shifts the dynamic and responsibility of testing to the team with the QE being in a player/coach role. The more accessible we make our tests (automated or not), the easier the adoption of their states will be.Some other characteristics to consider in the TAAS (Tests as a Service) ap[roach are:

- Concentrated in specific areas of the code
- Centralized (meaning anyone on the team can run them)
- Apply it to Regression testing
- Tests as a Service allows SWEs to validate on their own

9 Right #4 Leave No Trace

The Leave no Trace approach came about as it applies to camping. The idea is to leave a campsite or similar in at least the same condition as it was when you arrived. And, ideally, to leave it in a better state than it was when you found it. As we engage with the TAAS approach, this approach also applies! The idea here is to create, implement, execute and report on testing activities and offboard to the point where the team has everything they need to expand the tests and maintain them rather than have one individual do so. This approach is also very relevant to companies who have adopted an SDET or hybrid centralized testing approach. It also supports lateral movement across teams when specialized skill sets (like test automaton) are in demand or in a deficit. Some characteristics of the LNT (Leave no Trace) approach are:

- Leave things in a state as if you weren’t, or more importantly, can’t be there
- Onboard: assess, confirm, build, and deliver
- Offboard: backlog, maintain, steady state, and leave

Follow Prime Time guidelines. I like to call the standards and consistent practice that should be applied and adhered to by anyone who is building and delivering tests as Prime Time. Things to consider when defining Prime Time for your organization are:

- Standards and coding practices are aligned with Dev’s
- Agreed to by your committee of technical QEs

Next steps and how to apply all of these wrongs and rights:

Where do some of these “wrongs” apply to you? And more importantly, where can you make them right?

• Start small – focus on a Smoke test first and gain traction. Once you have a good adoption and review rate amongst the team, then expand automation.

• Tie it to Definition of Done/Shippable. This is a great collaborative item to bridge between Product and Engineering.

• Think like a team member – even if you’re not embedded

• Your end goal should be to provide self-service tests for anyone on the team to use- that is ultimately maintained by the team so you can focus on innovation driven by change and right the wrongs!