The "Do Nots" of Software Testing

Melissa Tondi

melissa.tondi@gmail.com

Abstract

In this talk, we will discuss some of the lessons learned in software testing – otherwise known as the “Do Nots”. Melissa will present the top five items that are practiced by and within QA teams where, at some time may have added value, but have now become stale or, in some cases, impactful to both the industry and project team. She will suggest different approaches and recommendations to help you either remove those items and replace them with more meaningful approaches or modernize them to ensure innovation is a driving force within your testing team.

Biography

Melissa Tondi has spent most of her career working within software testing teams. She is the founder of Denver Mobile and Quality (DMAQ), past president and board member of Software Quality Association of Denver (SQuAD), and Professional Services Manager/Consultant at Rainforest QA, 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 five “do nots” of software testing. These “do nots” 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 QA 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 quicker and with high quality. Here are the top “do nots” I’ve gathered in hopes that our community can continue to continuously improve and broaden our information sharing.

2 On-boarding for New Hires Only

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 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 Assume you have an Equal Seat

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 Assume QA/QE Owns all Testing

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.

5.1 Sample Automation Definition across Teams

We have 1-3 Priority criteria that aligns with the success of the agile team; not just QA/QE. This feeds in to each team's Definition of Done playbook and allows for more consistency, less ambiguity, and greater predictability to ensure teams are operating at the highest efficiency as possible.

- Development: P1 Intake Test
    - The ISTQB Definition
        - A special instance of a smoke test to decide if the component or system is ready for detailed and further testing. An intake test is typically carried out at the start of the test execution phase.
    - Intake tests are executed upon every code check-in and merge.
- Quality Assurance/Engineering: P2 Smoke Test
    - ISTQB Definition
        - A subset of all defined/planned test cases that cover the main functionality of a component or system, to ascertaining that the most crucial functions of a program work, but not bothering with finer details.
    - Smoke tests are executed upon merge and deployments and failures of these tests are designated as critical and would cause a HotFix if released in to Production. These tests are the baseline of the regression suite.
- Quality Assurance/Engineering and Product Management: P3 = User Advocacy
    - User Advocacy tests are tied to key customers or end-to-end flows and – anything that would cause a Priority 1 issue classification from Product Management or the Business.

6 Stagnant Missions and Offerings

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. Prove it!
- Report the outcome 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.