Shifting the Test Role Pendulum - Balancing Technical Acumen and User Advocacy

MELISSA TONDI

CAST 2016

My Background

- 15+ Years in Quality – Software Testing, Quality Assurance, and Software Quality Engineering - with 8+ years in management
- I work at ShopAtHome.com where I'm the head of SQE (previously QA)where we've successfully implemented all of the items following
- Before SAH, I was with Verizon/AOL as an Agile Quality Practitioner where I helped develop a refined scrum methodology - using the experiences of software testing to advocate the message that "quality is everyone's responsibility"

Takeaways

- Why the Pendulum?
- Discuss the changing landscape and timeline of the role of testers – from Y2K to Present Day
- Identify factors that caused the QA/Software Tester role to become mainstream in companies and what caused the pendulum to shift
- Fill in the gaps with a QE Model that addresses both technical acumen and user advocacy – emphasizing efficiency and time-saving strategies
- Present recommendations you can take back to your team to ensure a good balance between the two

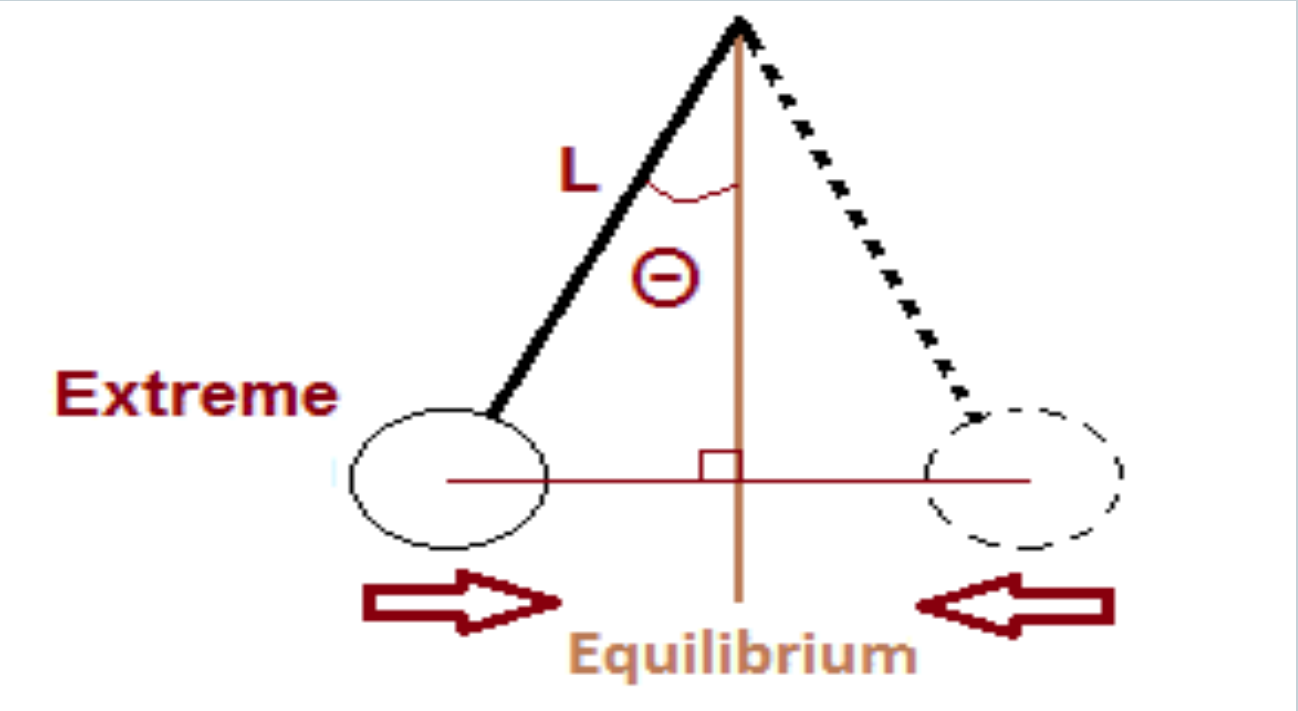
Testing – the Changing Landscape

Company Terminology Level-Set

- Mainstream Company – one that creates software products using technology (programming languages, tech stacks, etc.) that has been created by a…
- Bleeding Edge Entity – Microsoft, IBM, Google, Facebook, Apple. Versus…
- Late Adopters – Those reliant on legacy hardware, mainframes, highly regulated sectors.

Testing – the Changing Landscape

A pendulum is subject to a restoring force due to gravity that will accelerate it back toward the equilibrium position. A pendulum wants balance!



Testing – the Changing Landscape

Timeline for Mainstream QA/Test Teams

- 15-20+ Years Ago – Early on, Test/QA Teams were usually staffed from within the Business or End Users of the internal products being developed. The pendulum laid heavily on the side of the User.
  - In addition to a nominal understanding of the SDLC, the main skill sets needed were having product or domain knowledge and being detail-oriented
  - We emphasized heavy documentation (test plans/cases, strategies, scenarios) over actual testing
  - Some factors that contributed: Y2K, Wide adoption of the Internet

Testing – the Changing Landscape

Timeline for Mainstream QA/Test Teams

- 7-15 Years Ago – The influx of commercial tools and the mis-conception that these tools were a silver bullet solution glided the pendulum past the middle briefly, but gravitated to the Technical left.
  - Some factors that contributed: SaaS, heavy investment in commercial tools and therefore roles for specializations in those tools

Testing – the Changing Landscape

Timeline for Mainstream QA/Test Teams

- Present- 7 Years Ago - The Software Engineer/Developer in Test (S/DET), Mobile Tester and Automation Engineer roles have inundated the open Testing positions and the pendulum shifted mostly to the Technical left.
  - We have tended to put more emphasis on Development skill sets versus Testing skill sets
  - Our new moniker became: “In order to test code, you need to write code”
  - Lack of training or emphasis on the profession of QA/Test – “any developer can test”
  - Some factors that contributed: “Mobile First”, Big Data, etc.

Filling in the Gaps – An Efficient Operating Model

Automation – The Gateway to Efficiency

Exploratory/Ad Hoc – Increases with the Above

Accessibility – Introduced as a Result

Mobile – Begs to Become more Efficient

Performance – Shifts the Pendulum Right

Security – Highly Efficient

A Successful Automation Strategy

"Test Automation makes humans more efficient, not less essential"

A Practical, Three-Priority Strategy

1. What are the tests that are executed the most? Smoke/BVT/ Sanity. Automate them first!
2. Automate the Acceptance Criteria! Why?
   - This Ensures that it is agreed to and understood by the team
   - It is the most important piece that determines a release's success – MVP!
   - It can serve as the demo to the Product Owner
3. Dealer's Choice – Encourages collaboration across the team
   - Which tests are most important to Development? Pair!
   - Which scenarios are revenue or uptime critical?

Rules of Thumb:

- If a test is executed more than once, it should be a "first" candidate for automation
- Automation criteria should be defined by the discipline, but understood by the project team
- Not everything can or should be automated – leave that decision to the practitioners on the team. In this case, whoever executes testing tasks

Exploratory and AdHoc Testing – Just Do It!

"The Greatest Common Denominator (GCD) Approach"

Breaking down important characteristics into their most basic elements

- Tests are either:
  - Scripted or Unscripted
- Scripted Tests are executed either:
  - Automated or Manually
- Unscripted Tests are either:
  - Exploratory or Ad Hoc

Accessibility – Getting Started

Worldwide Web Consortium (W3C)
(http://www.w3.org/TR/WCAG20/#guidelines)

- You'll want to determine which Level is appropriate for your organization and users (details here), but here's the gist:
  - Level A – A good place to start. Great for organizations that already have a product in use and who want to establish a baseline for accessibility conformance.
  - Level AA – The next step. This level means that most people will be able to use your site/product in most situations. Many education and government agencies require this level.
  - Level AAA – The most difficult to achieve and maintain. In rare situations, this may be required, but the W3C makes it clear that it is not possible to satisfy all Level AAA success criteria for some content.

Accessibility – Testing the Guidelines

- Guidelines Verification Process. Our teams use the W3C checklist to create and execute tests, first manually, then by using a screen reader.
- Screen Readers
  - Window-Eyes
  - JAWS
  - WebAnywhere
  - ReadSpeaker
  - Fire Vox
  - ChromeVox

Mobile – Testing on Physical Devices vs. Sims/Ems

When to use Simulators/Emulators

- Testing the Function
  - Broken buttons, missing images, correct formatting for vertical and horizontal modes
  - RWD or mobile site elements display correctly
- Previewing the Form
  - Demo basic design and layout mockups before significant development investment is spent on polishing the UI
  - See the application within the frame of a real device, compared to the window of a development program
- Accelerating Development Time
  - For simple visual checks, opening an app in a device emulator can be as simple as switching windows, compared to fully loading a physical device
- Test more frequency with quick checks on an emulator, thus finding bugs faster and with greater context

Mobile – The Device Matrix Technique

Determining your Device Matrix

- Operating System
  - OS customizations, missing libraries, driver issues
- Screen Size
  - Rendering issues, usability, missing layouts
- Pixel Density
  - Density Independence, missing layouts
- Aspect Ratio
  - X,Y calculations, overlapping panels, display issues
- System on a Chip (SoC)
  - Hardware performance, Instruction set, battery, signal
- Carrier
  - Network protocol, speed, responsiveness, packet loss

Mobile - The Device Matrix Technique

Determining your Device Matrix – The Goal

- Efficiency, not necessarily 100% coverage
- Build your lab to cover both apps and mobile web
- Define a list of categories of devices
- Adjust the number of devices based on reasonable coverage

Augment When Needed:

- Perfecto Mobile
- Keynote
- AWS Device Farm
- Cloud Test Lab

Mobile - The Device Matrix an Example

Category	Device Name	OS	Size	Density	Resolution	DPI	AR	Protocol	System on Chip
Newest	Sam. Galaxy S	4.2.2	Normal	xhdpi	1920x1080	441	16:9	LTE/GSM	Qualcomm S4
Flagship	LG Nexus 4	4.2	Normal	xhdpi	768x1280	318	3:5	GSM	Qualcomm S4
Oldest	HTC Tattoo	1.6	Small	ldpi	320x240	143	4:3	GSM	Qualcomm S1
Popular	Sam. Galaxy S3	4.1.2	Normal	xhdpi	1280x720	326	16:9	CDMA	Samsung Exynos
Common	Mot. Droid 3	2.3.4	Normal	hdpi	960x540	275	16:9	GSM	Cortex A9
Abnormal	LG Optimus VU	4.0	Large	xhdpi	1024x468	256	4:3	GSM	Nvidia Tegra 3
Budget	Dell Venue	2.2	Normal	mdpi	480x800	228	3:5	GSM	Snapdragon S2
Catch-all	Sony Xperia P	2.3	Normal	hdpi	960x540	275	16:9	GSM	Sony NovaThor

Automation, Performance and Security

Flaws in our Current Approach

- These Engineers are generally not embedded with the team
- There is usually a hand-off from the Testers to these teams versus collaboration throughout

Performance Recommendations

- Embed– Bring in the the Performance Engineers as early as possible. If that's not possible:
  - Use Open Source Tools
    - Apache Jmeter (http://jmeter.apache.org/)
    - Grinder (http://grinder.sourceforge.net/)
    - loadUI (http://www.loadui.org/)
    - OpenWebLoad (http://openwebload.sourceforge.net/)
    - OpenSTA (http://opensta.org/)

Performance Recommendations

- Consider these Tests
  - Load – How does the system behave with a large number of users and what is the response time?
    - Log response time as a single user and feed that information to the Performance Engineer
  - Capacity/Scalability – How many users can the system support while not exceeding maximum page times?
    - Scale down number of users to typical usage and inject that within the functional tests

Performance Recommendations

- Stress– How does the system behave in extreme conditions?
  - This is usually not a good candidate, but be creative!
- Soak– How does the system behave over a long period of time?
  - Check for degradation with a small subset of users (use open source)
  - Account for periodic processes and run tests during that time:
    - Backups
    - Third Party Exports

Security

- OWASP (https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)
  1. Injection – Verify interpreters separate untrusted data from the command or query. **Intermediate**
  2. Broken Authentication and Session Management – Verify user credentials and session IDs are protected. **Advanced**
  3. Cross-Site Scripting (XSS) – Verify user-supplied input is properly escaped. **Advanced** except to verify safe JS APIs are being used.
  4. Insecure Direct Object References – Verify all object references have appropriate defenses. **Beginner/Intermediate** for validating direct object references are safe.

Security

5. Security Misconfiguration – Environments should be configured identically. Deploying updates and patches (including code libraries). **Beginner/ Intermediate**
6. Sensitive Data Exposure – Verify sensitive data is encrypted and that autocomplete is disabled on forms collecting sensitive data. **Beginner**
7. Missing Function Level Access Control – Verify the application restricts function level access via the UI. **Beginner**

Security

8. Cross-Site Request Forgery (CSRF) – **Advanced** except for CAPTCHA checks.
9. Using Components with Known Vulnerabilities – **Advanced**
10. Unvalidated Redirects and Forwards - **Advanced**

Summary

Takeaways

- The Changing Role of Testers
- Recognize the Factors that Cause the Change
- Identify Areas that can be more efficient
- Fill in the Gaps!

- Automation
- Exploratory/AdHoc
- Accessibility
- Mobile
- Performance
- Security

Recommendations

Contact Information

- email: melissa.tondi@gmail.com
- Twitter: @melissatondi
- Blog: MelissaTondi.blogspot.com
- LinkedIn: Melissa Tondi