Shifting the Test Role Pendulum

MELISSA TONDI

SQUAD2015

My Background

- 15+ Years in Quality – Testing, Quality Assurance, Software Quality Engineering with 8+ years in management
- I'm now with Verizon/AOL as an Agile Quality Practitioner where I'm developing a refined scrum methodology
- Avid fly fisher and (trying to be) golfer
- But, enough about me… who are you?

Takeaways

- Discuss the changing landscape of the role of testers
- Identify factors that caused the QA/Test role to become mainstream in companies and what shifted to where it became more technically focused
- Fill in the gaps with a test strategy/that emphasizes user advocacy tests
- Present recommendations you can take back to your team to ensure a good balance between the above

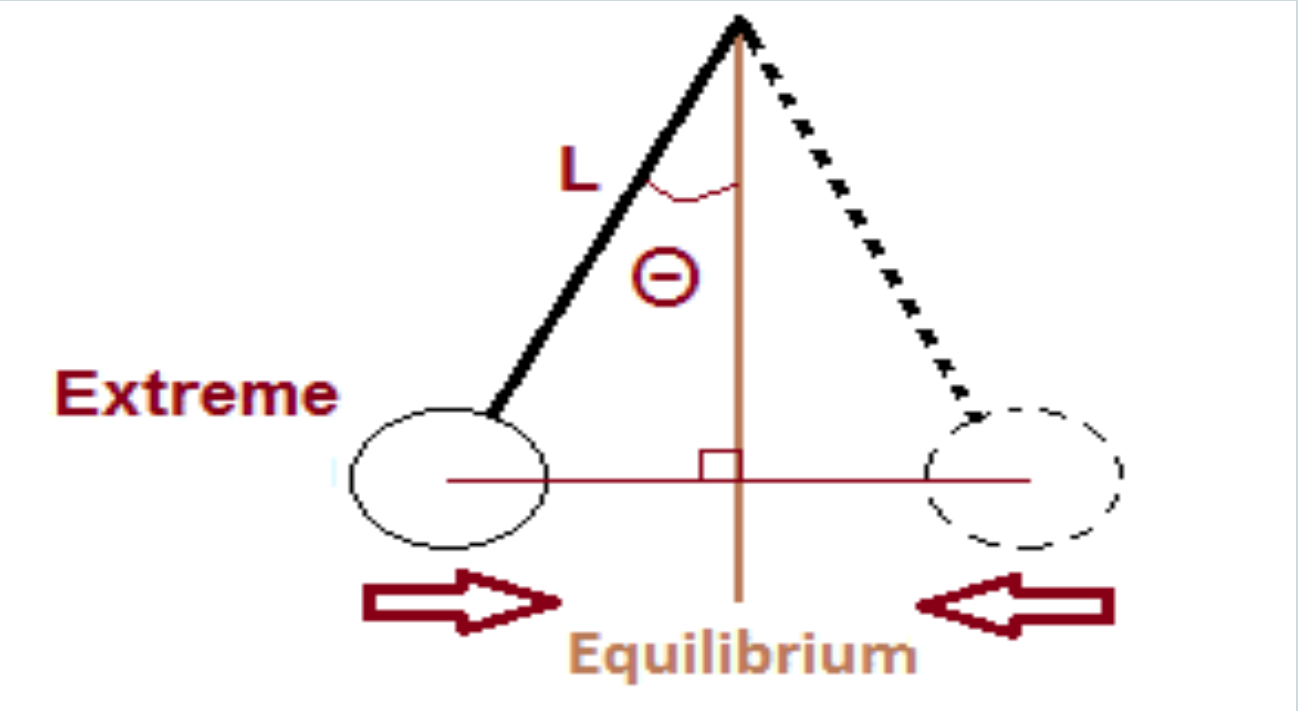
Testing – the Changing Landscape

Timeline for Mainstream QA/Test Teams

- 10-20+ Years – 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 healthy understanding of the SDLC, the main skill sets needed were having product or domain knowledge and being detail-oriented
- 7+ Years – The influx of more releases and the mis-conception that automation was a silver bullet glided the pendulum past the middle briefly, but gravitated to the left.
- Present - The Software Engineer in Test (SET) job description has inundated the open Testing positions and the pendulum resides mostly 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

Testing – the Changing Landscape

- In our efforts to hire high quality Test teams, has the pendulum swung too far away from the user?



Factors that Defined the Test/QA Role

- 10+ Years – Y2K, Internet was widely used
- 5+ Years – SaaS, Mobile, BYOD
- Present -

Filling in the Gaps

- Accessibility
- Mobile
- Automation
- Performance
- Security

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

- **Peripheral**
- Wireless Testing – NFC (Near Field Communication), Bluetooth/ Bluetooth LE Accessory, Stylus
- Wired
  - Internal to Device – Headphone Jack, Keyboard
  - External to Phone - CC Readers, Bar Code Scanners
- **Connection Testing**
  - USB Power/Data
  - 4G/LTE/CDMA+/Wimax
  - 3G/GSM/CDMA, 2G/Analog, WiFi, Hotspot Generation
  - Computer Tethering
  - Carrier
  - Throttle – Limiting bandwidth and measuring an App's performance

Mobile

- **Gestures**
  - Swipe, tap, pinch/expand, shake, orient, tilt, press and hold, swipe and hold, eye pause.
- **Interruption Tests**
  - Controlled - Plug in/out USB, power, and headphone, Home Button, Power Button, Navigating to another App, Save State
  - Uncontrolled/”Elevator” test - SMS, Phone Call, Notifications

Mobile

- **Syncing**
  - Updating two or more locations to ensure applicable files are equivalent.
  - App to Cloud, App to Computer, App to App, Device to Device
- **Internal Hardware Integration**
  - Camera, GPS, Accelerometer, Battery Drain, SIM Card, Volume, SMS, Microphone, Speakerphone

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

Automation

- Start small, but be consistent across teams
  - What are the tests that are executed the most? Smoke/BVT/ Sanity. Automate them!
    - 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
  - 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

Automation

"Automation makes humans more efficient, not less essential"

- It's not a replacement of humans, but in some cases there may be a change in responsibilities in the Testing team
- It is not a silver bullet – it won't immediately increase productivity, but if approached correctly, it will eventually support increased releases, greater test coverage, and overall quality of the product

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 User Advocacy Tests that may be Missing

Recommendations

- Accessibility
- Mobile
- Automation
- Performance
- Security

Contact Information

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