Turning a Marathon Runner into a Sprinter: Adopting Agile Testing Strategies and Practices at Microsoft

Jean Hartmann

Test Architect

jeanhar@Microsoft.com

Overview

- Embracing Change
- Quality-related themes
- Benefits & Challenges
- Q&A

Embracing Change…

- Past release cycles - 3 years or more
- Need more frequent release cycles
- Change - a perfect storm
  - New direction => devices and services
  - New development philosophy => agile
  - New org structure => federated model
- Extensive planning was needed

Key Quality-related Themes

- Moving quality upstream
- Increasing code velocity
- Adapting cross-platform testing
- Leveraging telemetry
- Integrating release criteria
- Ensuring rock-solid tooling

Highlights: Moving Quality Upstream

- Supporting a better developer (unit and component) testing experience
- Need to catch bugs earlier and reduce defect leakage
- Closely linked to componentization and refactoring effort
- Taking a practical approach to agile testing (not TDD right away)

Highlights: Moving Quality Upstream (2)

- New classes and components are being mocked and unit tested thoroughly
- Legacy classes/components are being refactored and tested where possible
- Devs and testers often share the workload
- Benefit : discussing product testability issues
- Anticipating that system level regressions will be reduced

Highlights: Increasing Code Velocity

- Supporting increased code velocity and deployment with confidence
- Need more efficient and effective pre-checkin validation
- Leveraging smart regression test selection strategies
  - Code coverage and churn based (dynamic selection vs. retest-all)
  - Want a focused and relevant test set with good coverage
- Piloting with product teams
- Anticipating faster and more confident check-ins

Highlights: Improving Product Testability

- Part of product componentization and unit testing discussion
  - Clean model-view separation within products
- Driven by bad system (UI) testing experiences in the past
- Devs and testers now want more controllability and observability!
- Anticipating:
  - Easier and more efficient testing of application logic/ UI components
  - Deeper, internal testing of application logic, e.g. app states, attributes, events

Highlights: Adapting Cross-Platform Testing

- Supporting more efficient and effective cross-platform testing
- Previous approaches resulted in multiple, platform-specific test suites
- Little sharing of tests or test environments was possible
- Costly to adapt to new devices and platforms

Highlights: Adapting Cross-Platform Testing (2)

- New approach covers improvements in test case design and test environment
- Applicable to native and browser-based apps (or both)
- Teams are authoring C# platform-agnostic system tests (repro-step-like)
- Teams executing system tests against multiple endpoints (devices)
- More agile by quickly validating shared, common code on new platforms and devices

Highlights: Leveraging Telemetry

- Advocating more and better product telemetry
- Few telemetry mechanisms available in the past
- Need to reduce product downtime, improve customer experience
- Online services led the way – now client products are following!
- Unified telemetry and flighting infrastructure is being built
- Looking at impact on agile testing processes
  - Focusing on keeping test suites fresh and relevant

Highlights: Integrating Release Criteria

- Includes accessibility, world-readiness, security, privacy, etc.
- Improving analysis, collection and reporting practices across development lifecycle
- Past efforts were costly and time-consuming, but tolerated due to longer release cycle
- Improving agility by:
  - Automating more tasks, e.g. static/dynamic code analysis tools
  - Reviewing and streamlining design and coding standards, practices and workflows
  - Consolidated reporting of criteria and aligning with top customer experiences/scenarios
- Attempting to make this workload an integral part of a sprint backlog

Highlights: Providing Rock-solid Tools

- Often overlooked - good test tool and reporting support remains important!
- New tools need to be deployed, existing one streamlined and everything well-integrated to improve overall workflow
- Test cases/scripts need to run faster, catch more with higher reliability
- Examples:
  - **Unit test harnesses** that enable Devs to run their portable, in-proc unit/components tests across multiple devices and platforms
  - **System/scenario test tools** that enable testers to author and run cross-platform tests across multiple devices and platforms
  - **Leveraging virtual machine technology** to run tests more efficiently cf. physical machines
  - **Providing better reporting 'dashboards'** to monitor test case health

Benefits & Challenges

- Devs are now more aware and intentional about producing quality code
- Dev and Test disciplines growing closer, better interactions on key topics
- Pace of innovation is picking up, e.g. cross-platform/smart test selection
- Teams are gaining better/more insights into how customers are using products
- Tooling deficiencies are being exposed and remedied faster

- Keeping going at this faster pace and improving all the time…

Summary

- Chronicled our Office journey – turning the marathon runner into a sprinter
- Highlighted key test-related themes
- Reflected on the benefits and challenges

- And hoping to get here soon!