QA or the Highway 2016 Presentation Notes

Making QA Strategic - Let's Get Real (Panel Discussion)

- Does testing belong at the strategic table?
    - What is that strategic value that testing provides?
        - Conquering that space of the "unknown" between releases
        - Testing can advise on the process and capabilities of the system
        - Delivering this information to the senior levels
        - The traditional value of QA has become less relevant as Agile practices have increased visibility
- Yahoo eliminated the QA organization, too much?
    - Testers need to share what they do to the organization
- How is testing viewed in your organization?
    - Used as gatekeepers (sign-off on Release)
- Is the Yahoo way, the right way - In Agile it is everyone's responsibility to test?
    - Systems Thinking (Quality Software Management - Chapt. 1)
        - Product Level Risks
        - Team Delivery Risks
- What other skills do we need to develop in addition to Systems Thinking?
    - Just In Time - build partnerships with those you work with (including Customers)
- Testing is not creating software, just criticizing - What value is that?
    - Looking at the change logs (commits) rather than the ticket - better idea of what is really changing and uncovering hidden changes -> Risk -> Change logs, version control
- Outsourcing, What about it?
    - This is a periodic cycle
    - Supplementing, not replacing with outsourcing
    - Focusing on costs actually raises costs (Failure Demand)
- How can we educate the powers that be of our value as Testers?
    - Lunch and learn talks, tell to everyone in the organization
    - Test teams have a better understanding of the overall system
    - Testers leading customer testing and 2nd tier support
- What about Automation? Automating ourselves out of job?
    - Automators may not deliver the same quality as manual
        - Point out Key Bugs/Finds
    - Balancing Manual and Automation
        - The Swiss Cheese of Risk
        - Coverage is overall managing of risk
    - Instilling quality from Day 1 (focussing on the development stage)
        - Focusing on the unit testing with pairing with QA (Cross-Functional Testing)
        - Automation up to through the API (Integration), then manual testing for GUI
- Are these Skills and Values only for Agile or can it be applied to Waterfall?
    - Waterfall, no real idea of what is going on until it hits test
    - Continuous Integration

    - We could have always done these things, Agile just gives us “permission”
    - We must get better at making the status of the system more visible
- What bad habits do we still have?
    - Adversarial relationship with Dev, lessening under Agile
    - Just focusing on finding the bugs,
        - Tester provides information on the state of the system (Information Brokers)
- What are the top 3 things that you can do to provide value
    - Stay focused on the customer and the capability that your software is providing (why are we building this thing)
        - Partnership with the customer (using same tools as the customer e.g. UAT testing tools)
        - Be easy to communicate with
    - Innovation - look for ways to inspire innovation
    - Responsibility to stay up to date with current testing practices and thoughts and community
    - Modelling personas
    - Gap Analysis
    - Look for hidden opportunities to deliver value
    - COMMUNICATION including asking questions
    - Try new things, new ideas
    - Finding ways to get other people in your organization to find out what you do, and find out what they do

The Limits of Unit Testing and How to Exceed Them

- Why QA software?
    - Health and well-being of our co-workers and the user
- What kinds of bugs do you want in your final, QA approved product?
    - None, if we cut the system down enough
    - Most software has way too many features
    - MVP
        - What must your product do (or don't do) or you don't have a product
        - Protect mental and physical health of users and co-workers
        - What else must be true for the 2 above to hold
- The quality plan drives the tools and technique not the other way around
- Manual testing - How do human being respond to the software that we build
- Parse Int.
    - Is this for trusted or non-trusted input?
    - Can I trust that my function will be invoked correctly?
    - What is the culture of the input?
- Bad Error Handling Kills
    - 92% catastrophic failures a result of bad failure handling
- How can I be completely confident in a simple function
    - At least handles failures correctly

Unit tests are...	
great...	**not so helpful...**
● Think thru bottom-up designs ● Preventing regressions ● ???	● Showing overall design consistency ● Finding Security Holes ● ???

- How can I be completely confident when composing two functions
    - should produce the correct results
- State Space: Input produces output, but output may be influenced by other factors than just the input
    - database
    - web
    - time of day
    - random number generator
- Formal Methods
    - Context-free grammar (programming language)
    - Errors in the scope and knowledge of the specifications
    - Regular Expression maps to a Finite State Machine (legitimate states make the testing tractable)
- Property Based Testing

    - For some random input there is a property that must hold for that output
        - Reversing a list twice returns the original list
- Fuzzing
    - AFL - American Fuzzy Lop?
        - marks up the binary of your application
        - starts with an input corpus and creates more from it especially if it finds one that breaks the target, can takes weeks to run
        - resource intensive
        - great at finding crashes but not incorrect results
    - Fuzz testing
        - assertions to check
        - specification of what the fuzzer should inject into the system
- Combine Fuzzing with Property Based Testing
- Use other systems to verify our system
- Presenter is developing system called Fizil combining several of these techniques

Agile Test Evolution: Changing Test Strategy over time

- What are the requirements?
- Subject Matter Expertise in the domain
- Patterns of problems as companies mature
- Agile Principles, how do these change testing
    - Deliver working software frequently….
- Scrum
    - The flow problem (10 stories, all pulled by dev on the first day, then on the 10th day they all drop into Ready for Testing)
    - Bottleneck - constrains flow
    - QA is not the problem or constraint, just last in the process
    - Scrum doesn't tell us how to solve this
    - What about regression testing?
    - *Beautiful Testing* book
    - What about pushing regression of Sprint n to start of Sprint n+1 (Parallel Pipeline)
        - Accordion Effect (too much work in progress, too many branches)
    - Failure Demand
        - Demand that is accidental (call centers, outsourcing the call center, but now the person answering the call can not solve your problem, so you call back so now you have 2x transactions at least so no cost savings)
        - When your sole focus is reducing costs for knowledge-based industries, it actually costs more
        - Preventing defects that we can - initial quality (fix, retest, repeat)
        - Measure Failure Demand and reduce it
    - Building the system while it is up and running
- Extreme Programming (the book)
    - Manual testing, don't (for institutionalized, scripted, repetitive testing)
    - Jared Small at Blackberry and their personna development
- Testers are a mini Product Owner (or not)
    - ATDD, BDD, & The 3 Amigos (George Dunwoody)

        - Everyone wants to do a good job (Dev, Tester, Product Owner)
        - A combined, meaningful understanding of what the system should do
        - Specification by Example book on how to have these conversations
- GUI Tests - checking the change, not testing
    - Fitness (tool), Check the API, Poke the GUI
- Reducing the regression test window or just specify what you are testing
    - Low tech testing dashboard - MAKE THE STATUS OF TESTING VISIBLE then let others decide if it should go out using information
    - Develop a custom strategy for this release
    - U-Test (another line of defense)
- Reduce Work in Progress / Limit WIP - Slack
- Component Strategy
- Continuous Deployment - The Swiss Cheese Model of Risk: you can have holes in your risk strategy but you can't push a pencil all the way thru
    - Not suitable for all domains (medical devices)
    - Less regression testing, more testing for emergent risk
- Testing is growing

A Paradigm Shift in Quality Engineering

- Things are REALLY changing around what testing really is
- Does it still make sense to consider one's self as just a tester
- SDETs & Test Engineers
- *How Google Tests Software* (book)
- No Testers Allowed (Microsoft, Facebook, Yahoo)
- Are testers and endangered species?
    - in some places, yes
- Old Release Cycles (with infrequent releases, important not to have bugs)
    - Bugs were immortalized
- Daily releases are soooo yesterday
- Agility pushed us into this direction
- Automation Tools
- Competition breeds innovation
- Software is getting better at built in quality (and even auto recovery)
- Quality vs Suitability (Idea bugs are devastating)
    - Building the right product vs building the product right
    - "Production Quality" some bugs are OK, get to market
- Slow is Dead
    - manual is slow
    - bottlenecks are slow
    - Ownership Shift - developers taking control and responsibility for quality
    - Devs will make stuff testable if they have to test it
- Facebook
    - Sheer size of user base
    - "Dog-fooding" (use your own products)
    - Lock in, only game in town
    - A/B testing with limited number of users 1% is 16M users

        - Users are doing the end-to-end, black-box testing
- Google - Engineering Productivity Group
    - Recognizes that there's a difference in Mindset (Testers)
    - Looking at code and how to make it testable
    - A threshold tho before devoting time to tests
- Microsoft
    - Combined Engineering (everyone is a developer)
    - Benefits:
        - Developers who don't have to worry about quality build houses of cards
            - So for some groups, quality went up
            - Leaner organization moves faster
    - Issues:
        - "Brain drain"
    - Adjustments
        - Software developers learning new things
- What do we do now
- Roles combine
    - Want everyone on team to be able to do everything (maybe not as well as others but…)
    - Understanding each other's domain (how to fix flakey tests? why is it flaking?)
- What of QA?
    - Orchestration
- What's your world?
    - Facebook (rapid releases) to rocket science (better get it right)
- Become the shift (Own the automation if the work you do can be automated)
    - New Skills
    - Testing has few limits
- One sizes does not fit all

Lookup

- F# programming language
- Idris programming language (based on a proof system), not yet ready for prime time
- Zen Hypervisor vs. ? Hypervisor
- AFL
- QA Automation System Canopy
- Conference: Star East
- TestOps
- Telemetry (what are your users doing and how are they doing it)
- “Bug Bash” (Testing swarm)
- www.ministryoftesting.com what do testers do
- www.watirmelon.com