Essential Patterns of Mature Agile Testers

This webinar focuses on sharing the 13 patterns testers need in order to move from "Doing" agile to "Being" agile. Bob Galen and Shaun Bradshaw draw on their vast experience as agile Practitioners to provide real-world examples for each pattern.

The Differences between Agile Testing vs. Traditional Testing:

- Traditional testing focuses on just testing while Agile testing has a quality focus
- Traditional testing relies on detailed requirements and documentation while Agile testing focuses on team interaction for requirement clarification
- Traditional testing is plan-driven while Agile testing uses minimal test plans
- Traditional testing silo's test teams by domain and technology while Agile testing relies on higher competency across domains and technologies
- Traditional testing relies on Test Management tools and Big "A" automation tools

"Doing" vs. "Being" Agile

Testers need to move toward "being" agile, rather than just "doing" agile. "Doing" Agile focuses on tactics, ceremonies and techniques. "Being" Agile focuses on team mindset, leadership mindset, behaviors, organizational adoption, etc.

The 13 Patterns:

1. Ruthless KISS (Keep It Simple Stupid)

- Exploratory testing is done by the whole team
- Try not to do duplicate tests at different levels
- Start trusting baseline tests that are being done
- Don't try to do it all. We are trying to do just enough, but the right just enough
- Focus on what is most important at that time
- Notion of Trust
    - In Waterfall, there wasn't trust. In agile, you have to trust everyone

2. Swarm to the top

- Avoid “Scrummer-fall”
    - For many agile teams, there is 1 sprint dedicated to requirements, 1 sprint dedicated to development, and 1 for testing. This is just time boxed Waterfall. Even if you have testing at end of 2 week sprint, it is still Scrummer-fall.
- As a team, focus heavily on the most important stories and tests
- Do your analysis and do exploratory testing on the fly
- Test plans and test cases will be much more minimalist
- As soon as the code is ready, go ahead and start your testing
- Test Strategy and Test Plan should be at the Release Level
- Each sprint, focus on getting the test cases executed and finding and retesting defects

3. Whole Team QA Ownership

- Everyone is responsible for testing/quality
- Opportunistic pairing between Developer and Tester
- Engage in debate
- Design for test and testability
- Focus on today for what are the most important tests
- Stop reinforcing thinking in terms of silos (“Development Complete” or “Test Complete”)
- Everyone on the team owns the QA outcome

4. Quality on ALL Fronts

- Quality has to be built in
- Shift focus from **defect detection** to **defect prevention**
- Shifting Left in our thinking
- Alter team’s mindset and actions from I shaped to T shaped
    - I-shaped people do only the things that they are good at (For example, developer develops)
    - T-shaped person has core expertise but can do other things outside their general skillset. For example, I am a developer but if push came to shove, I could write automation scripts, or I could create acceptance criteria.
    - Need teams to have T shaped people.
    - I shaped people lead to silo approach
    - Everyone has one strength but we are trying to expand ability beyond that skill.
    - Bring whole self and be flexible.

- Can't test in quality.
- Trust developers to do unit testing
- Focus on Craftmanship and Professionalism

5. Active Done-Ness

- Done-Ness tends to have heavy focus on Development activities. As a tester, make sure that you include writing test cases, doing pair code reviews, putting tests in repository.
- If automation is important, don't say that you didn't have time.
- QA tasks have to be included in estimates and criteria for done-ness.
- Think in terms of what has to be done as a tester and incorporate that into your estimates.
- There are many scrum and agile instances where teams don't have a definition of done, or a lightweight definition. Active done-ness has a robust definition of done.
- Levels of Done can be defined at Task, Story, or Sprint level in Agile, or by Release for SAFe.

6. Communicate Early and Often

- Face to Face, User Stories, Sprint Planning are all parts of communication
- Embrace idea of 3 Amigos: Persona of Developer, Tester, and Product Owner
    - They need to talk about the stories.
    - Amigos talk from inception to Done
    - Amigo mindset does not leave out tester
    - All 3 perspectives matter
- There needs to be active communication outside of stand up.
- Don't wait until the stand-up to mention an issue. If there is blocker/impediment, communicate it to the team as soon as possible.
- Mindset is every day is special. Maximize every day.
- Real time communication, make adjustments as we go.
- Depending on conversation, there may be need to pull in additional expertise (DBA, UX or Architect)
- Face to face communication is the best way to address complex issues
- **Write less, talk more**

7. Continually Engage the PO

- Product Owner needs to be your new best friend
- Product Owner needs to understand and describe the "why?" behind the stories
- Why drives how we do our testing
- We can help formulate the acceptance criteria if we understand the why

- Product Owner owns the priority which drives our testing
- For defects, talk to PO for priority
- Product Owner is the voice of the customer and helps us understand the value proposition
- Testers need to partner with the Product Owner to solve the customer's problems.

8. Build Trust with the Developers

- Trust that developers can test their own code
- Build cross-team trust
- Mindset of letting go of archaic views that we have (Tester vs. Development)
- Trust that team members will do the right thing or ask for your help

9. Test Case Failures – What if it's not a bug?

- Difference between really good tester and bad one is what he or she does when test case fails
- Be cognizant of the time limitation to address bugs.
- How tester reacts to a defect is really important
- Just because a test fails, is it really a defect? Verify the following first to be sure that you really found a bug:
    - Can the failure be duplicated?
    - Was the test properly executed?
    - Were the test steps correct?
    - Was the failure a result of environmental or data issues?
    - What were the error messages?
- Doing some additional test exploration may drive out additional information that might help developer to fix the issue.
- Have conversations first and document second
- Get up and talk to developer and team
- Conversation can drive out so much more context
- Don't write up defect and throw over the wall. (That is waterfall mentality which will alienate development and reaffirm silos.)

10. Agile Test Automation- aka Flip the Triangle

- Historically, Automation Testing was focused on testing everything at UI level. If you couldn't get there from UI, you couldn't automate.
    - Middle and API couldn't be tested
    - Pricey tools were used for automation
    - Small changes to code could break tests

    - Rework to test was often necessary

- Changed mindset and strategy in order to flip the triangle and invest most of time in Unit Tests.
    - Less Middle tier tests and even less UI tests
    - Unit tests run really quickly
    - Effective strategy-The agile test automation strategy
    - Number of tests and tools are up to you… by level
    - Organizations pick different framework for each level. That is the trend
    - Defacto standard in agile teams is to concentrate most on unit tests

11. Continuous Learning: Yours + Team

- Learning is continuous. You don't just read SCRUM guide and get certification and then stop learning.
- Identify some regular learning goals and break them up by Sprint
- Think in terms of Shu Ha Ri
- Find a mentor and/or set up a Community of Practice around Agile, Test Automation, or Testing

12. Yes, There is Planning in Agile

- Apply Risk- Based Testing techniques at a daily, Sprint and Release level
- What did we think we were going to deliver today? Do we need to make a pivot?
- Test plans are important. They are planned as a team with tester taking a lead.
- Planning as a team is everything

13. Metrics, i.e. What to Measure?

- Metrics are not about an individual. They are about the team as a whole
- Velocity is the most common metric
- Flow-How quickly do things make it thru?
- Throughput-How much gets thru your sprint?
- Predictability-How often do you hit your measurements?
    - What is your standard deviation on that?
    - Predictability matters to Product Owner
- Changes to definition of done?
- How often do the stories slip?
- 80% done dilemma
    - Team has routine of just getting 80% done-We want to meet our commitments.

- Also need to look at quality of product from customer using it and organization that we work through.
- Is it bringing an actual dollar benefit? More money in the door? Are customers happy with the quality?
- Track team happiness.
    - Happy teams are better for organizations.