Welcome!

Take a minute think about the way testing happens on your team....
Forget the QA!
We’re a Cross-Functional Team!*
(*But who has time to test anything?)

Julie Clooney & Marianne Erickson
The Culture and Mindset of Software Testing

Has any of these things ever happened on one of your teams?

• As a money-saving measure, QA Analysts are shared across multiple departments.
• Developers are expected to test their own code as part of the development process, and that completes the test plan.
• Your management says your team doesn’t need to test your software if you build it right in the first place.
• Your team really means to test, but up against deadlines, there simply isn’t time to complete the test plan so they just hope for a miracle.
• The team runs the test scripts, but they really don’t seem relevant to the application or its users.
• Tester(s) reported one or more issues but the software was deployed anyway because there wasn’t time to fix it.
That place... in case you hadn’t noticed, is....

TESTING HELL
Redemption is possible...

Before we talk about the path out of Testing Hell, let’s examine how we got there....

*The road to hell is paved with ineffective testing practices....*
Types of Testing

- **Basic**
  - Black Box
  - White Box
  - Grey Box

- **Phase**
  - Unit
  - Integration
  - System
  - Acceptance

- **Type**
  - GUI
  - Functional
  - Validation
  - Security
  - Smoke & Sanity
  - Regression
  - Monkey
  - Gorilla
  - Agile
  - Usability
  - Exploratory
  - Ad-Hoc
  - Static / Dynamic
  - Compatibility
  - Stress / Load
  - Performance
  - Volume
Categories of Testing

**Functional**
- Unit Testing
- Integration Testing
- Smoke / Sanity
- Localization
- Globalization
- Interoperability
- So on...

**Non-Functional**
- Performance
- Endurance
- Load
- Volume
- Scalability
- Usability
- So on...

**Maintenance**
- Regression
- Maintenance
Making The Shift

Shift Left

- Test For Prevention
- Test Every Stage
- Test Continuously
- Test Early
- Test Every Time
- Test Everything
Best Practices

- User Acceptance Testing
- Operational Readiness Test
- Requirements Verification

Test Strategy
- Automation Strategy
- Test Schedule
- Resource Planning

Delivery

Test Dev
- Test Plans
- Test Matrix
- Test Scripts
- Test Data

Defect Management
- Bug Tracking
- Bug Fixing
- Bug Verification

Test Execution
- Defects
- Test Reports
- Test Metrics
### Elements of Agile Test Strategy

<table>
<thead>
<tr>
<th>What</th>
<th>Why</th>
<th>Where, When</th>
<th>How, Who</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category &amp; type of testing, e.g., “functional acceptance”, “performance”, or “exploratory”.</td>
<td>The kind of requirements that this type of testing is intended to address, such as “acceptance criteria”, “performance requirements”, “error handling”</td>
<td>Where: The environment(s) in which the testing will occur. &lt;br&gt;When: The events or frequency that will trigger the testing start and end.</td>
<td>How: The method to be used for those types of tests, such as “Use Jbehave / Java, Selenium.” &lt;br&gt;Who: Who will do what: Who will write the tests, who will perform them, etc.</td>
<td>Strategy for measuring code coverage / traceability to acceptance criteria. &lt;br&gt;What level of coverage is sufficient (e.g. 95% coverage).</td>
</tr>
</tbody>
</table>
### Example Testing Strategy

<table>
<thead>
<tr>
<th></th>
<th><strong>Unit</strong></th>
<th><strong>Acceptance</strong></th>
<th><strong>API/Service</strong></th>
<th><strong>System / Regression/ UAT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Why</strong></td>
<td>To Ensure Code is Written Correctly</td>
<td>To ensure customer’s expectations are met</td>
<td>To ensure communication between components are working</td>
<td>To ensure the whole system works when integrated</td>
</tr>
<tr>
<td><strong>Who</strong></td>
<td>Developers / Technical Architects</td>
<td>Developer / SDET / Manual QA</td>
<td>Developers / Technical Architects</td>
<td>SDET / Manual QA / Business Analyst / Product Owner</td>
</tr>
<tr>
<td><strong>What</strong></td>
<td>All new code + re-factoring of legacy code</td>
<td>Verifying acceptance tests on the stories, verification of features</td>
<td>New web services, components, controllers, etc.</td>
<td>Scenario Testing, User flows and typical User Journeys, Performance and security testing</td>
</tr>
<tr>
<td><strong>Where</strong></td>
<td>Local Dev + CI (part of the build)</td>
<td>CI / Test Environment</td>
<td>Local Dev + CI (part of the build)</td>
<td>Staging Environment / QA / UAT</td>
</tr>
<tr>
<td><strong>When</strong></td>
<td>As soon as new code is written</td>
<td>When the feature is ready and unit tested</td>
<td>As soon as new API is developed and ready</td>
<td>When Acceptance Testing is completed</td>
</tr>
<tr>
<td><strong>How</strong></td>
<td>Automated, Junit, TestNG, PHPUnit</td>
<td>Automated / Manual</td>
<td>Automated, Soap UI, Rest Client</td>
<td>Automated / Manual</td>
</tr>
</tbody>
</table>
Roles & Responsibilities

Team Members
- BA
- Dev
- CR
- QA

Scrum Master
- BA

Architect (ARCH)

Product Owner
- PO
- QA

Development (Dev)
And What is a Whole-Team Approach to Testing?

A Whole-Team Approach can include several elements that are familiar to Agile teams.

1. **Test Driven Development (TDD), Behavior Driven Development (BDD), and Acceptance Test Driven Development (ATDD)** all facilitate the team in finding direction toward testing.

2. **Test automation, whenever possible, frees team members for manual testing**, so keep the manual testing relevant – don’t waste hands on keyboard for something than can be automated.

3. **UX and UI testing** remind the team of why, and for whom, we are creating applications in the first place. If you aren’t comfortable using the application, you end user won’t be either.

**WARNING**: A Whole-Team Approach to testing is NOT based on cost-avoidance and does not make professional testers irrelevant.
These things are a good start!

- Bring new agile testers up to speed quickly, but don’t overwhelm them
- Clarify testing activities within the team (each team is self-organizing)
- Collaborate with business experts to identify valuable features and deliver the right capabilities
- Design automated tests for reliability and easy maintenance
- Facilitate ways for Agile team members to improve and expand their testing skills
- Plan “good enough (ish).” Balance small increments with larger features and the entire system.
- Use testing to identify and mitigate risks and prevent defects
- Quickly address challenges within your product or organizational context
- Perform frequent exploratory testing using “personas” or real users
- Engage the whole team in your exploratory testing approaches
Testable User Story

A good user story should be:

Independent (of all others)
Negotiable (not a specific contract for features)
Valuable (to the business or end user)
Estimable (to a good approximation)
Small (least amount of work to deliver value)
Testable (in principle, even if there isn’t a test for it yet)

As a...<"CUSTOMER" or "USER" who directly benefits from the successful delivery of this User Story>

I want to...<perform a “FEATURE” specified in the User Story>

So that...<"BENEFIT" / "VALUE" that the customer or user will enjoy on the successful delivery of this User Story>
Clear Acceptance Criteria

Given
What software will look like to user

When
Things that the user will do

Then
What the user should expect
1. Provide continuous feedback.
Think about the word “continuous!” Don’t save feedback for the end of the line (or the last day of the sprint). Start testing as soon as possible!

2. Deliver value to the customer.
Even if the customer (or end user) can’t see the value, that doesn’t mean value isn’t there. Enhanced security adds value. Speed adds value.

3. Enable face-to-face communication.
If you are one of the lucky few who has customer testers (real UAT), by all means, meet them in person whenever possible! Issues are easier to explain when the tester or end user can demonstrate.
(Agile Principle 6)

4. Have courage.

5. Keep it simple.
Test what is, not what could be or should be.
Ten Principles for Testers

A decade ago, Lisa Crispin and Janet Gregory published their Ten Principles for Testers. These hold true more than ever!

6. Practice continuous improvement.
   Do you talk about testing in your retrospectives? If not, start now!
   (Agile Principle 12)

7. Respond to change.
   (Agile Principle 2)

8. Self-organize.
   The best team approach to testing is the self-organized team approach.
   (Agile Principle 11)

9. Focus on people.

10. Enjoy.
    Testing is fun! How often are you allowed to try to break things?
Implement these ten principles on your team (and in your team charter), and you are well on your way to whole-team testing success!
Workshop Time!!

1. Stand up and stretch!
2. Check under your chair for the golden ticket!
3. Gather into table groups of 5 to 9 people to a team, and prepare yourself to think!

Trivia!
Why gather in groups of 5 to 9 people?
For This Exercise

• Who is going to test what?
• How are they going to test it?
• Where and when will it be tested?
• What processes / methods or tools will be used?
• How will you track your coverage (how will you know when you are done?)
How did it go?

Workshop Debrief

A couple of things always happen when a full team owns the testing process:

• Pride of ownership – the product now belongs to the entire team.
• Increased accountability – the “no defects on my watch” syndrome.

What happened on your teams?
Switching Gears

What happens when no one on a team accepts accountability for the testing process?

- Who has ever flown on a Boeing 737?
- Did anyone receive this, or a similar travel advisory, before you flew here this week?

Travel Advisory

737 MAX 8 Aircraft Update

In compliance with the Federal Aviation Administration’s (FAA) order on March 13, 2019, our Boeing 737 MAX 8 aircraft remain temporarily grounded—until further notice, you will not be flying on a 737 MAX 8.

Because the MAX 8 represents less than five (5) percent of our daily operation, we expect a limited number of Customers to be affected by its temporary removal from our fleet. If a Customer is not satisfied with their new travel plans, they can rebook on alternate flights without any additional fees or fare differences between the original city pairs.

The limited number of Customers who have already booked their travel and will be affected by our amended schedule are being proactively notified of their re-accommodated travel plans, according to our flexible accommodation procedures.

The Safety of our Customers and Employees is our uncompromising priority, and we remain in communication with Boeing and the FAA. We will resume regular MAX 8 operations when authorized to do so and will continue updating you as we know more.

We’re sorry for any inconvenience experienced as a result and appreciate your patience as we work to accommodate our affected Customers. You can check your flight status at Southwest.com.

To see ongoing updates and accommodation procedures, visit our MAX 8 Updates page.

- When no one is accountable for testing, and know one even knows WHO is accountable, then in extreme cases, people could die. Specifically, 346 people could (and did) lose their lives.**
Building Quality In

High Level Scenarios
- Analyze the story (PO, DEV)
- Scenarios creation
- Scenarios validation (PO)

Detailed Scenarios
- Study of story’s details (DEV)
- Test cases creation

Automation Test Creation
- Functional tests creation
- GUI tests creation
- Add to Build

Test Execution
- Unit tests
- Verification
- Functional tests execution
- GUI tests execution
- Manual tests execution

Final Validation Sign-off
- Exploratory Testing
- Performance Testing
- UAT Testing (PO)
UX and UI: The User-Centric Tester

We write USER stories for our end users, and we should keep our end users in mind when we test, as well. As we stated earlier, if you don’t like testing your application, your users are going to like it even less!

Is this a smooth User Experience?
Is this a smooth user experience?
UX and UI: The User-Centric Tester

For what user was this designed?

Do you think it was tested on very many actual users?

Was USABILITY the key to this design?
Does The Design Work For The User?

**Description**

France is a contemporary bathroom toilet that embraces the latest trend in luxury modern bathroom design. The ultra-modern one piece toilet features a sophisticated water-saving dual flush system and S-Trap design.

The sleek rectangular modern bathroom toilet features smooth edges and bowed back drop. The curvy rectangular bowl portion of this chic Italian style modern bathroom toilet masterfully joins the moderate tank in perfect unity. This fashionable one piece dual flush toilet design marries beauty with contemporary functionality adding comfort and elegance to any modern bathroom decor.
How do you test for the end user and not for the stakeholders?

Our requirements come from where?

Is that the End User?

How DO you test for the end user?

- Personas
- Think of your grandmother!
- Test as someone who doesn’t know the application!
- Use the Gherkin scripts!

**Given I am a new member without a completed medical profile**  
**When I select prescriptions to refill**  
**Then I am prompted to complete my medical profile before Submit Order button displays**
What Contributes to a Whole-Team Approach to Software Testing?

Don’t be shy! Shout it out!

Inclusive Collaboration

System Testing

TDD/BDD

Regression Testing

Requirement Specification

Test Validation

Test Set-up

Test Execution

Output Comparison

Fault Mitigation

Reports

CROSS TRAINING

TEST AUTOMATION

UNIT TESTING

Daugherty BUSINESS SOLUTIONS

Confidential & Proprietary to Daugherty Business Solutions.
In Summary – A Whole Team Approach

- Includes a test strategy that focuses on delivering value to user
- Keeps the Ten Principles in mind
- Evolves
- Enhances team accountability for the quality of the end product
Websites:
Modern French/Italian Dual Flush Extended Toilet. The Interior Gallery: www.theinteriorgallery.com

Books:

NOTE: The Three Pillars are:
• Development and Test Automation
• Software Testing
• Cross-functional Team Practices

Other Resources:
Ministry of Testing. [https://www.ministryoftesting.com/](https://www.ministryoftesting.com/)

**The Boeing 737 MAX Saga: Lessons for Software Organizations, Phillip Johnston And Rozi Harris Vol. 21, No. 3/© 2019, American Society for Quality**
Thank you for joining us today!