The traditional view of testing and the traditional role of tester has changed immensely with advent of recent trends in testing like Agile, Mobile, Cloud, Globalization and others. Its not alien if i say many of the teams have confusions on what has changed and how do we adapt to the new trends. Agile being one of them, it would be great to put my thoughts forward in six simple steps on what should be encouraged in the testing process to ride one of the testers perfect storm that has been written a while ago.

Since a decade businesses have seen tremendous competition and this competition is to continue if not with a bit more aggression. This definitely brought the much necessary change in project teams to become lean in all respects with every effort linked to business value. While development teams continue to progress in such an environment with ease, the role of testers is still being experimented and explored. This article talks about the challenges and best practices for such agile test teams.

Key Challenges In Agile Testing

- A very limited documentation on requirements and more reliance on collaborative meetings to brainstorm should we term "Stories"
- An accepted view that code delivery will be incomplete by the time testers start their testing
- Changes accepted right until the last minute and at times even middle of development
- Much more responsibility in terms of being part of and driving stakeholders to define requirements, driving teams to ensure testable code delivery.
- Identifying the right balance of test procedures between manual, regression and Automation testing

6 Best Practices To Embrace Agile Testing

Exploratory Testing

Traditional structured testing has always focused on writing test scripts, reviewing test scripts and executing scripts in a sequence and hence always ended in expected behaviours. However the demands of agile based delivery means a simultaneous learning, test design and test execution which is termed "Exploratory Testing". With exploratory testing, testers always look for emergent behaviours that could not have been predicted otherwise.

Exploratory Testing consists of following

- Learn, Design, Execute tests simultaneously
- Look for Emergent Behaviours that expose value to stakeholders
- Explore the system by running high level tests, take outputs from these tests and device yet more powerful tests
- Test a part of system, if not many bugs found either test other part of system or change test techniques to test the same part
- Ask not "Does application meet these requirements?" instead ask "What happens if i do this?"

Embrace The 5 Values

The traditional development and testing always relied on robust documentation as proofs, a divided project team in terms of stakeholders, developers and testers and focus on just a specific area as per the persons role. The success of any agile implementation or the scrum framework itself is standing on five core values .

- Focus
- Openness
- Commitment
- Courage
- Respect

The teams are expected to be focused on their tasks and are committed to their completion. They are quite open and honest in their thoughts and expressions, have courage to challenge themselves and the others, having the respect for one another at the same time.

Stories Should Be Right "INVESTment"

Unlike before test team are expected to play their part right from the start of project driving the requirements process, helping and challenging development teams in terms of estimates and other areas. It is absolutely necessary that team as whole come up with requirements or user stories that adhere to following principles and are right investment for right value in return.

- **I** - INVEST
- **N** - NEGOTIABLE
- **V** - VALUABLE
- **E** - ESTIMATABLE
- **S** - SMALL
- **T** - TESTABLE

Demand Test Driven And Continuous Integration

The very important aspect of an agile development or testing is "Test Driven Development" and "Continuout Integration".

Test-driven development (TDD) requires developers to create automated unit tests that define code requirements (immediately) before writing the code itself. The tests contain assertions that are either true or false. Passing the tests confirms correct behavior as developers evolve and refactor the code.

Continuous Integration is small pieces of effort to integrate source code at very regular intervals within a day by the development team with a view to avoid complexity of integration at the end, ensure good quality of code and reduce time taken to deliver it.

Testers have a great role to play in terms of demanding TDD/CI at every stage of development to ensure a quality deliverable comes out of development. Test team have to help generate an automation framework that could be integrated into TDD and CI process to reduce the complexity of development and testing process.

Regression Tests Are The KEY Test Artefacts

The ability to accept change at any given point within agile delivery mandates a finite set of tests that can test the whole system in principle at any given time. If you examine carefully without proper requirements documentation, test plans and with nature of testing being exploratory "Regression Tests" are the way to go to provide confidence to self, business and the team to say an area of system being delivered is enough tested within the aggressive sprint schedule. Most of the agile teams embraced regression as an opted strategy for testing and when executed on a continuous basis without failures definitely provides product stability.

Regression tests provide a basis to take decisions like below which otherwise would be hard to answer using manual testing in a 2 week iteration

- Is the code ready to be merged?
- Is the product ready to be released?
- Has this change broken something previously working?

Automation, Automation and More Automation

Its an established fact that automation is the only way to generate testing capacity to do a complete test of a release or a shippable code on a regular basis across any standard iteration. With less available testing time, have an ability to accept changes at anytime, needing to drive the team in terms of TDD i guess the only way to be able to fulfill such a repsonsiblity is to generate time on the long run by automating tests as much as possible and as early as possible.

Automation helps

- Accelerate Velocity of Agile Teams (Agile demands speed)
- Consistency in Testing (Across Environments and Various Stakeholders)
- Identify Issues Early On With less effort across any build

A sufficient amount of governance should definitely be in place for test automation and regression to ensure teams dont end up having a complex and volumnous tests to be run every time on the long run.