Best Testing Practices in a Lean Agile Environment

Continuous Testing requires being focused on providing value for the business. Below are industry-standard best-practices for testing in an agile environment.

- Reduce unnecessary testing artifacts, such as extensive test plans and test cases
- Adapt a more exploratory attitude to testing when testing manually

Collaborate with Business

Continuous Testing means testing correctly from the very start. We have to make sure we get good requirements from business to start development.

- QA should build a close relationship with Business Analysts.
- Remove ambiguity from user stories - ensure every user story is testable and includes acceptance criteria
- Every process has precise entry and precise exit criteria
- Unit tests defined during the design process
- Functional specs force clarity from the perspective of a designer and architect
- Don’t ignore non-functional testing such as performance and security. Do both functional and non-functional testing from the very start of the project.
- Build meaningful end-to-end test scenarios by utilizing trends, data and analytics from the production website to gather information about user activities and user journeys through the application.

Implement a Test Practice

- Build a strong testing practice which drives development.
- Create robust test scenarios (valid, invalid and edge cases) in feature files
- Identify actions to be taken when failures occur
- Consider Usability - the final arbiter of quality
- Provide feedback on methods to improve the user experience
- Test product on different platforms
- Release to a limited number of customers for beta testing
- Gain insight on how product meets needs of customer

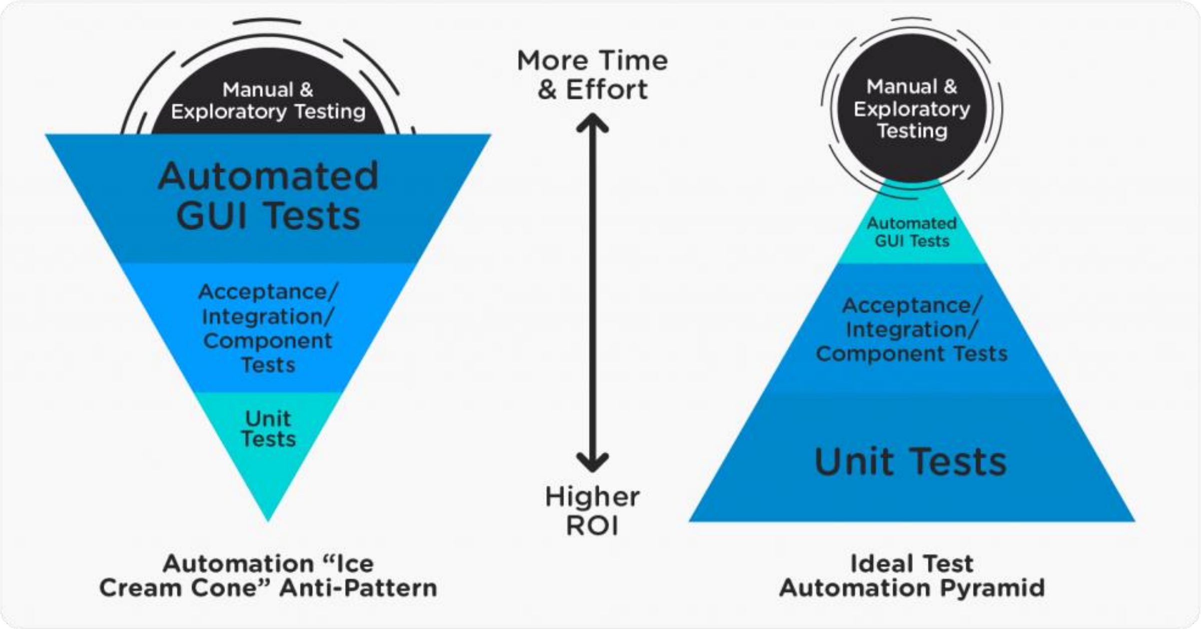
Implement a QA Practice

- Run regular QA workshops where the testers can improve their technical skills as well as soft skills.
- Implement appropriate Test Techniques, leveraging technical architecture diagrams, models of the application and mind maps.
- Embed QA within the entire teams so that they are aware of any changes to the application.

Automate Testing

Continuous Testing requires testing early and testing often. We can use automated testing to get quick feedback on the status of the application.

- Apply Best Practices on Test Automation
- Know when to automate tests and when to leave them as manual tests
- Test automation is the responsibility of both developers and testers.
- Automate regression tests as well as non-functional performance and security tests where possible.
- Ensure you follow the Test Automation Pyramid principle by increasing automated unit tests, API and Integration tests, and only a handful of automated tests through the UI.


Manual & Exploratory Testing
Automated GUI Tests
Acceptance/ Integration/ Component Tests
Unit Tests
Automation "Ice Cream Cone" Anti-Pattern
More Time & Effort
Higher ROI
Manual & Exploratory Testing
Automated GUI Tests
Acceptance/ Integration/ Component Tests
Unit Tests
Ideal Test Automation Pyramid

- Run automated tests from a Continuous Integration (CI) server.

- Create smoke regression packs that run fast and run them as often as the application is updated.
- Automate new functionality and stories along development rather than leaving them for later.

Establish Quantifiable Metrics

- Measurement method that uses the defect stream to quantify product process
- Tools assist in this measurement and provide guidance on coverage
- Track results

Automate Deployment

- In order to make the most of continuous testing, the steps involved in every stage should be seamless, trouble-free and automated.
- Rather than waiting for DevOps to deploy the latest release in a test environment for QA to test, this process should be automated.
- Embrace **Task Automation**. Automation is not just for testing and verification. Any repetitive heavy-processed manual work should be automated.
- Standardize the test environments so deployment is smooth across different environments and the results of the automated tests are reliable.
- Make use of visualization to scale automated testing to get quick feedback.