DESCRIPTIVE ESSAY of testing terms – Simon Steinaa

The 4 agile testing quadrants

The 4 agile testing quadrant is a matrix diagram that gives an overview of how tests could be divided in an agile project. Each of the four quadrants reflects the different reasons we test.

The quadrants(1,2) on the left include tests that support the team as it develops the product.
The quadrants (3,4) on the right include tests that critique the product to make it even better.

The lower left quadrant(1) represents TDD with automated tests (unit and component). These tests are for programmers - not business experts - and focuses on internal quality.

The upper left quadrant(2) also support the work of the development team, but at a higher level and defines external quality and the features the customers want. This includes tests that describe the details of each story, functional tests. Mock-up and wire frames are used to help validate proposed designs with customers. These are manual but also automated tests that support the team to get the product built right.

The upper right quadrant(3) represents manual tests only to check that the requirements was met, because errors do occur, since it is human to err. These tests include: exploratory testing, scenario testing, user acceptance testing, usability testing, alpha/beta testing. This focuses on providing feedback to quadrant 1 and 2. It also helps the customer to test the system and find new ideas for missing requirements.

The lower right quadrant(4) represents tests that focuses on non-functional requirements: These include performance tests, load tests, stress tests, maintainability tests, scalability tests and security tests. These tests are performed by using tools. If the developers know the non-functional requirements before it is easier to design and implement with that in mind.

The Agile Testing Quadrants matrix helps testers ensure that they have considered all of the different types of tests that are needed in order to deliver value.

System testing

The system is tested as a whole to verify the requirements. This includes both functional(“what the system does”) and non-functional(“how well the system does it”) testing. It is a black-box testing technique, since it requires no knowledge of the inner design of the code or logic. Some defects that unit tests cannot expose can be revealed via system tests. Types of system testing can include scalability / load / performance / stress / reliability / functionality / usability / security testing.

Exploratory testing

Exploratory testing combines learning, test design, and test execution into one test approach. It uses the tester’s understanding of the system, along with critical thinking, to define focused, experimental “tests” which can be run in short time frames (time boxed , one or two hours) and then fed back into the test planning process. In order to do good exploratory tests, the tester needs good intuition and the skill to follow code ‘smells’. Before an exploratory test, a charter is first planned for the first ‘attack’ based on your current knowledge – then as soon as you learn more about the system – the charter is revised and new tests are conducted. Exploratory testing is good if there exists poor requirement specifications and documentation.