A **Defect** in simple terms is "a variance from expectation". A Defect is a condition in a process/product which does not meet a documented requirement. In other words, a defect is an error in a process or product's behavior that causes it to malfunction or to produce incorrect/unexpected results. The root cause of an incident may originate from different sources*: code, requirements, design, environment, build/compilation, test case, data, etc.*

This definition is purposely broader than the more common "variance from business requirements or system specifications". Software failures often originate in the analysis or design tasks of development projects. Requirements and specifications may be incomplete, inaccurate, or vague. By keeping the incident definition broad, the Development Team member may draw upon all of his/her knowledge of the customers' need to determine if a defect exists. In an agile development approach, team members contribute to all aspects of system development. They are involved in project scoping discussions, business requirements definitions, and specification design sessions. Through this participation, they acquire much knowledge about customer expectations that is never captured in a formal document. This knowledge enables team members to gauge quality -- that is, identify defects –- based on the standard of customers' need.

A **deferred defect** is any unexpected behavior observed when exercising the asset that <u>will not</u> be corrected during the release.

Another type of defect is an **escape defect**. They are unexpected behaviors that are detected after a Feature or User Story has been completed (i.e. they escaped detection during a sprint/cycle).

e. **Test Automation –**

Test automation will focus primarily on the Regression test cases. The strategy is to utilize and Object Oriented approach with Data-Driven Modular steps/methods. The technology to be employed for standard Regression includes Gherkin/Cucumber/Selenium/Java integrated into the Eclipse IDE. To expedite Regression executions, Sauce Labs virtual machines will be used to run all suites in parallel to minimize the duration of the Regression executions.

Where a test case doesn't lend itself to automation (i.e. the execution time is too long as compared to system timeouts) **<u>OR</u>** is not related to a repeating event (ex/ re-platforming an application), Manual testing will be performed using an <u>SDLC standard template</u>.

An additional consideration for 'automation' is to define **Continuous Test Integration** rules within the Jenkins tool such that the following will occur:

- Unit-level Regression testing will occur as each code module it checked into the repository
- Nightly Shakeout/Regression testing will occur for all functionality in the repository
- Shakeout testing will occur for all functionality when a build is defined and deployed.

These additional regressive test executions are being developed via our DevOps rollout. A "pipeline" (or collection of jobs/tasks) is being defined for our deployments such that each deployment task is followed by a Shakeout test (defined below). If the Shakeout is successful, the