BUG TRACKING PROCESS

The process varies depending on where a bug is found in the development cycle.

Generally, bugs will be tracked and fixed using the APM tool as the primary point of reference. This allows all members of the team to track all work in a single spot. But there are exceptions, as outlined below.

DURING AN ITERATION

Bugs identified within an iteration for which the story is being developed are tracked on the project wall using the following process:

1. Tester notes bug on stickie note and attaches it to the story card

2. Tester places a bug sticker on the story card

3. Tester moves card back to In Development and notifies responsible developer

4. Developer fixes bug and resubmits it to Ready for Test on the project wall

AFTER AN ITERATION

Bugs identified after an iteration where a story was considered "Done" must be placed on new stories and prioritized along with other development work by the Product Owner team.

1. Tester creates a defect story card in the APM tool and notifies the Product Owner representative about the defect card.

2. Product owner representative then:

    1. Schedules the defect for resolution

    2. Defers the defect for a later release

IN INTEGRATION TEST, SYSTEM TEST AND UAT

For bugs found during regression testing in System Test and UAT, defects must be entered into Quality Center for the support of quality metrics, departmental objectives and to facilitate defect resolution and quality transparency with dependent teams.

1. Tester must enter defect into Quality Center

2. Tester creates a defect Story Card in the APM tool and notifies the Product Owner representative about the defect Card.

3. Product owner Representative Then

    1. Schedules the defect for resolution

    2. Defers the defect for a later release

4. Tester updates status in Quality Center based on decision of the Product Owner Representative

DEFINITION OF TERMS

UNIT TEST – Tests created by developers testing small units of code. These are generally run during the Continuous Integration process. We will track code coverage and overall number of tests as core quality metrics for the development team. These tests will be developed using jUnit and standard extensions. Unit tests do not require the application be deployed to run

INTEGRATION TEST – Integration tests test the applications functionality from front to back and require that the application be deployed in Websphere to run. These tests validate the functionality for each story and will eventually provide automated regression tests. These tests are generally created by both the testers and development group. Several tools will be used for several types of integration testing including; Fitnesse, Selenium, jUnit and Quick Test Pro.

SYSTEM TEST – System tests are tests that are written in Quick Test Pro and run against the application in the System Test environment. Selenium tests may be run in the system test environment to validate Cross Browser Compatibilities.

ITERATION PERFORMANCE TEST – A small set of integration tests written in jUnit will be instrumented to validate Load and Performance criteria within the Continuous Integration server. These are not a substitute for overall L&P but allow the team to track performance numbers early in the development cycle.

PERFORMANCE TEST – Formal Load and performance testing run prior to a release. These tests are run as the current L&P process does.

ITERATION UAT TEST – Manual acceptance tests run by the Product Owner team at the end of each iteration.

UAT TEST – Traditional UAT test run each release. These are manual tests and represent the regression test for all functionality touched by the system.

TESTING TOOLS

Information on Fitnesse and Selenium is available in the tool profile documents

QUICK TEST PRO – Used for regression test of all functionality in the ST environment.

SELENIUM – The team will be utilizing selenium to test cross browser functionality for each of the supported browsers. Selenium supports many standard browsers while QTP only supports IE.

FITNESSE – Fitnesse will be utilized as a testing platform for validating the Web Service interfaces that we will be interacting with.

JUNIT – Standard unit testing framework for Java with extensions for testing J2EE Applications.

TESTING ENVIRONMENTS

BUILD ENVIRONMENT – Hudson build environment that runs all unit and jUnit integration tests. Builds check every 5 minutes to see if new code has been checked in. If there is new code Hudson checks out all the code and builds the application and then deploys it to the Development Integration server if all Unit Tests pass.

   * Hudson Build

DEVELOPMENT INTEGRATION – This server has the last good build of the application. This environment is available for developers to run basic tests against. There may be two instances of this environment as needed.

   * Devl2 – Developer Sandbox

ITERATION TEST – Hudson may be used to deploy the latest good build to the Iteration Test environment. This environment will be used for validation of functionality within the iteration and demoing the application at the end of the iteration. There may be two instances of this environment as needed.

   * Devl3 – Iteration Testing

Full integration and pre-production testing – From the Agile dev and test environments, code will be promoted into the SHARED IT, ST, UAT/PT environments. The tools typically used in those environments for quality management will be applied to the code from the Agile teams just as it is for everything else. This includes the use of Quality Center. As mentioned earlier, in addition to using Quality Center in the Shared environments, the team will also use the APM tool to prioritize its defects.

TOOL ENVIRONMENT MATRIX

-	Build	Integration	Iteration Test	System Test	UAT
jUnit	x	x	x		
Selenium			x	x	X
Fitnesse	x	x	X		
Quick Test Pro			X	X	X