Notes on the Test Management Tool (TMT)

Ian Fore

Reviewed 3/19/12

Definitions added 3/26/12

Consolidated material from other notes document 3/31/12

General observations 1

Requirements 2

Test plan 2

Some other notes on test plans 3

Questions that we want to be able to ask 3

What test scripts have been written for caTissue? 3

By who? 4

What tests have been run? 4

By who? 4

On what builds? 4

What is the test workload? 4

Definitions of entities in TMT 5

Test plan 5

Definition 5

Reality check 5

Actions 5

Test cycle 5

Definition 5

Reality check 5

Actions 5

## General observations

In general – tables suffixed info contain the actual details on the thing named*. E.g. scenario*\_info contains the name and other details of the scenario whereas the table scenario is a cross reference table to other data about the scenario.

The four levels

Product

Version

Component (though the table is called component\_info)

Test area

Is component the same as in Bugzilla?

Note that Version is not in any sense used for the notion of a software version. Just treat it as a set of folders.

There is then a many to many relationship between testarea and testcase. The component table manages that relationship. No idea why it is called “component”

Note that in prior versions of TMT (e.g. that from 2009) the primary key of testcase\_info was auto\_id. In the current version it has become identifier. Note however that some tables still use a column called auto\_id as a foreign key to testcase. Any time you see auto\_id it is safe to assume it is a link to test case.

## Requirements

Requirements as they are created are not associated with anything below the level of a product.

Requirements are associated with a test cases in the many to many relationship managed through tcinfo\_req. As part of that - a requirement indirectly, and after the event, can become associated with an area or component. Someone cannot sit down to write test cases for a given component/area and know which requirements are associated with that component/area.

## Test plan

The primary key of testplan used to be tpr\_id – now it is identifier. Tpr\_id has been retained as a (cryptic) foreign key to testplan.

Testplan\_info

Test plans have no association to product – it is therefore not easy to see which test plans apply to a given product.

Once it is decided to run a given set of tests it should be sufficient to create the requisite set of test\_result records. At the time of creation these represent actual tests that will be run.

How do you know what to run in each case?

When you create a test plan you select a set of test areas and that gives you a set of tests to run for that test plan.

This means that the flexibility to run different sets of tests is entirely dependent on how you set up test areas.

You can cherry pick (actually cherry remove) the test scripts for any particular plan by removing them from the list.

There is no capability to easily apply the same set of tests to different scenarios.

Nor is there the ability to run the same set of tests (plan) on different builds.

Test plans are being created for each build.

### Some other notes on test plans

As later testplans from the same “master” plan are created they include more tests presumably because the tests have been added to the test area in the meantime

Usage note – the “same” plan may include more tests at later times.

Test plan 769 – as an example

The only duplication of tests within this scenario is for test areas. There is only one scenario.

Does this mean the same test is run twice for the different areas? How would the test be different for the two different areas? The scripts are the same.

## Questions that we want to be able to ask

What test scripts have been written for caTissue?

By who?

What tests have been run?

By who?

On what builds?

What is the test workload?

## What test scripts have been written for caTissue?

The first problem is that there are a number of products whose tests are managed in TMT. And worse – there are several “products”/”versions” which are called caTissue in one way or another.

Query of products and versions

The question then becomes how do test cases get moved from version to version?

Count of requirements by product

|  |  |
| --- | --- |
| Prod\_Name | # of requirements |
| caTissue | 785 |
| caTISSUE Core | 63 |
| caTissue Core - LSD | 1 |
| caTISSUE Core\_migrated | 59 |
| CaTissue suite 1.2 | 1 |
| caTissue Suite-DFCI | 1 |
| catissue-LSD-CBIIT | 1 |
| caTissueSuite | 129 |
| caTissueSuite-Indiana | 126 |
| caTissueSuite-TJU | 61 |
| caTissueSuite-UPenn | 35 |
| caTissueSuite-UPitt | 27 |
| caTissueSuite-WU | 2 |
| caTissueSuite-Yale | 12 |
| caTissueSuite\_migrated | 151 |
| Catissue\_V1.1 P4.3\_Phase1 | 1 |
| Dynamic Extensions | 120 |
| pcaTissue | 1 |

Further research suggests that only caTissue is relevant, but it begs the question as to why other requirements are no longer relevant? What has been left out and why?

### By who?

## What tests have been run?

### By who?

### On what builds?

## What is the test workload?

# Definitions of entities in TMT

## Test plan

Table: testplan\_info

### Definition

A test plan is a set of tests put together for a particular purpose.

### Reality check

A test plan is of minimal value if the purpose for it is not clear. For example, two substantially different test plans exist for MAGE but it is hard to determine the intent of the person who created each of them.

### Actions

Ensure a description (column tp\_desc) is provided for all test plans that would help someone coming in without prior knowledge to know what the purpose of the test plan is.

## Test cycle

Table: testcycle\_info

### Definition

A test cycle is a particular set of test executions for a particular

A number of test plans may be executed during a

### Reality check

This is named a cycle because it is an instance of a thing that is repeated. The thing that is being repeated is a test plan – or set of test plans

### Actions

The build name/version should be in the cycle name.

Add a supplemental table which contains the list of builds. Populate it with the build names from Bugzilla.

Add a column to testcycle\_info to point to the build.