1. Introduction and overview

1.1 Purpose of this Document
This document describes how we will test our code for robustness. It includes test cases and other methods of testing.

1.2 Scope of the Development Project
The project entails creating an interface for The American Academy to sponsor tasks for employees to complete to help their company grow the teaching materials base. This will require heavy use of the Drupal Content Management System that TAA uses to create modules representing tasks, projects containing tasks, and the interfaces to store, create, and search tasks.

1.3 Definitions, Acronyms, and Abbreviations
Drupal - Content Management System for administering websites.
TAA - The Academic Academy
V&V - Verification and Validation Plan

1.4 References
SRS - http://www.eng.utah.edu/~blatnick/cs4500/SRS.pdf

1.5 Overview of Document
This document will describe ways in which we will debug, test, and verify our code as being accurate and free from defects. It will include ways to maintain bug-free code through frequent tests as well as considerations to keep in mind while developing and aspects to test that we may have misses in the production phases

2. Summary of Results
All the tests we have done so far have been successful except for the modules we have not yet finished. There seems to be a bug with the Views module which causes PHP to do a white screen. We will look into this.
3. Results from reviews, walk-throughs, inspections, and audits

Because Drupal is an Apache, PHP and MySQL development environment, we have been writing and developing on our own laptops to later upload to our sandbox at 71.195.228.8.

Our weekly meetings consisted of reviewing which direction we should take with our Drupal code, which modules could be implemented, and discussion of any possible alterations.

Because PHP is a type of language that is compiled, testing is usually done as soon as the code is written. PHP isn't gracious at letting us know when an error occurs however, so this real time testing/compiling is a bit more tedious than something written in Visual Studio.

4. Results from testing

4.1 Summary of component test

<table>
<thead>
<tr>
<th>Component</th>
<th>Pass/Fail</th>
<th>Explanation</th>
<th>things left to do</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drupal Core</td>
<td>PASS</td>
<td>Configurations worked correctly. The user module was able to successfully login. We removed irregular white screens.</td>
<td>Check schema and configurations after project is complete</td>
</tr>
<tr>
<td>Search Module</td>
<td>PASS</td>
<td>The search module (search and javascript display functions) are working correctly. The display the information that we request via views and search.</td>
<td>Must write a few more view accordion for different aspects (money owed etc)</td>
</tr>
<tr>
<td>Workflow Module</td>
<td>FAIL</td>
<td>We still need to work on the workflow module so that we can distinguish between tasks that are accepted, closed, over the expiration date, and under review.</td>
<td>We need to implement this</td>
</tr>
<tr>
<td>User Module</td>
<td>PASS</td>
<td>The user can login to Drupal Core successfully. The password is hashed in the database correctly. The user has an appropriate money amount owed to him.</td>
<td>Test when the workflow module is complete</td>
</tr>
<tr>
<td>Project/Task Module</td>
<td>PASS</td>
<td>The admin can create projects/tasks, and the user can view them. Once our workflow module is complete, the user will be able to accept and modify them as well.</td>
<td>Integration with the workflow module</td>
</tr>
</tbody>
</table>

4.2 Summary of integration test/testing product as a whole

We are working well toward our goal of having a complete task/project system for TAA. We need to implement the workflow module still, but most of the core components of this project are complete.
5. Evaluation of the process

The testing process included demonstrations of the current progress of the work with critiques from other team members. Due to the fact that we are still in the production phase of the project, debugging by other team members was irrelevant as the modules still need to be worked on. It was more effective to evaluate the progress so far and offer suggestions of features, interfaces, and improvements that the author of the module may have missed.

5.1 Evaluation of test cases

We found that our modules are still lacking in some structure needed for the scope of the project. Our "tasks" need to be enhanced to allow storage and retrieval of a client's individual progress in a task. Also, we need to discover what is causing our White Screens of Death (WSOD). Possible reasons for the bugs are closing php tags before the end of the file or compilation errors.

The Accordion Module worked well, as well as the representation of the tasks. Functionality is still missing, but we are making progress and the layout is there.

Test cases for the accordion module could have been concise, but perhaps we could have added more test cases for the tasks and the module to view a list of tasks. Due to the nature of web development, the test cases basically involved clicking around the interface looking to break things and handling bad input.

5.2 Results from defect tracker

Existing Bugs and Missing Features:
* "White Screens of Death" - Reason: unknown
* Change behavior from a 3rd party module using Drupal Hooks within our own module. This includes removing some unnecessary output and adding some features.
  * Store Task State on a user basis
  * Integration between tasks and viewing tasks (right now the viewer uses a generic task representation).

Error output from apache concerning the White Screen of Death:
[Wed Mar 11 12:06:01 2009] [error] [client 127.0.0.1] File does not exist: /home/steve/Site/drupal/sites/all/

We have reduced the number of White Screens of Death successfully through removing a closing php tag from a module. In drupal, white screens can result from compile errors as well as closing php tags. This happens because if there is any whitespace after the tag, then Drupal is unable to redirect on an error since headers have already been sent to write white space to the buffer of text sent as html.

5.3 Lessons learned

Knowing what we know now, we may have considered focusing our attention on implementing the tasks more before working on the task viewer, because changes on tasks influence the way we search them. However, since our search functionality is modular, it shouldn't be a significant problem to expand the functionality to fit the prototype tasks.

The goal to keep in mind it to keep our project as modular as possible, because TAA's needs could change
over time and by being modular we will also allow our code to be divided easier.

6. Outcome of acceptance test and delivery

The user has the ability to log in and can view the tasks. Administrators also have the ability to create new tasks. We are close to having the content displayed in a satisfactory way, but we still have work to do in this area. The workflow is the area that’s the most lacking. We don't yet have the ability for administrators to accept completed work or pay the worker. We also have no ability yet to customize workflows. We don't yet have a product we feel comfortable demoing to TAA. We plan to have this in two weeks.

6.1 Acceptance Test Summary

Components tested:

• user (teacher) can register with the site and obtain a login name and password – PASS
  expectation: Every iteration we will check to see that the user can log in and out.
• user can view tasks - PASS
  expectation: Every iteration we will check to make sure that tasks can be viewed
• user can accept tasks - FAIL
  expectation: We are working on implementing the task application module. It should be done by Tuesday.
• user can complete tasks - FAIL
  expectation: This is part of the task application module.
• user can be compensated for tasks - FAIL
  expectation: This is a module we are working on part of the workflow module.
• user can only see tasks for is clearance level - PASS
  expectation: With our new modules, we will check to make sure this is still working.
• admin can create tasks - PASS
  expectation: This should be working fine with nothing further required.
• variety of prices - PASS
  expectation: This works easily along with the tasks
• variety of titles - PASS
  expectation: This works easily along with the tasks
• variety of multilevel task trees - FAIL
  expectation: This may be axed for the final version, will get TAA approval.
• admin can cancel tasks - PASS
  expectation: This works easily along with the tasks
• admin can delete tasks - PASS
  expectation: This works easily along with the tasks
• admin can compensate user for completed task – FAIL
  expectation: This is a module we are working on part of the workflow module.

7. Summary of open issues

Most of the needed modules are still in construction. The current issues with these modules are still developmental bugs which should mostly be cleared up once the modules are completed. Once the primary modules are created the next issues will be making sure they integrate together correctly and completing some of the optional enhancement modules.
8. Additional information