



High Level Description	
Description	Testing Strategy
System(s) Impacted	Strategies
Document Type	Reference Guide
File Reference	Psshr://Cms/Training, Doc & Support/~Administrative Documentation/Reference Guides/RG_Testing_Strategy.doc

Revision Control			
Date	By	Action	Pages
10/11/2004	C Medders	Document created	All
10/13/2004	P Siegel, M Radisch	Document reviewed and approved	All
10/13/2004	C Medders	Minor grammatical changes made throughout	All
11/07/2005	C Medders	Document revised	All
11/08/2005	B Drussel	Changes to test script section	3
11/08/2005	C Medders	Added What to Test section	3-4
02/29/2008	A Everett	Document converted to accessible template	3
05/01/2009	A Everett	Document edited	3
06/09/2009	J Reeds	Document reviewed	

Table of Contents

Overview	2
Testing Phases	2
Testers	2
Types of Testing	2
Test Scripts	4
What to Test.....	4
Reporting Problems	4

Overview

This reference guide explains the San José State strategy for testing CMS baseline upgrades and releases, including updates, fixes, patches and PeopleSoft bundles. Testing is a critical element to the successful maintenance of software and is necessary to ensure the following:

Items are delivered/fixed as indicated in baseline documentation.

Users can still perform all tasks (changes in documentation may be required).

Local customizations are not impacted by baseline changes (changes to pages, processes and reports may be required, as well as changes in documentation).

Testing must be completed in a thorough and timely manner to ensure that ample time is available for local changes to be made as necessary and to log tickets to baseline as required for items that may not be working.

Testing Phases

Unit Testing

Unit Testing is the first step in a major testing process. During this phase, users test functionality in the software to ensure that pages exist, processes can be run and data can be entered. Testing is typically done in a silo-like fashion, and there is typically no integration testing done during this phase. For example, an HR tester might create a new position, but the position wouldn't be used in any other processes. The test would be to ensure that a new position could be created.

System Testing

System Testing is the second step in a major testing process and typically what's done when testing smaller releases, updates, fixes, etc. This phase of testing is intended to test business processes and how data moves from one module to the next. For example, an SA tester might test the load of applications from Mentor, move those applications through the staging tables, evaluate them and matriculate them. Another tester might take over and test maintenance on those same applicants as they become students and move through the various processes a student goes through.

User Acceptance Testing

User Acceptance Testing is the final step in a major testing process and is typically the last set of testing done before go-live. This phase of testing mimics system testing in that all processes and integration points should be tested. Testers should test errors and other issues that were identified during System Testing to ensure they have been corrected.

Testers

The functional leads should identify one primary and one alternate tester from each functional area. Testers should be well-versed in the areas they are assigned to test and should have no problems navigating the software.

The CMS subject matter experts (SMEs) will assist and participate in testing as required, but the primary onus of testing is on the functional representatives. Technical staff will be assigned to test nightly loads and other jobs kicked off in batch processes. Communication about testing plans will be done through the functional leads, and they will be responsible for ensuring the areas they oversee get tested. In all testing phases, testers will be required to sign-off that they have completed the various test scripts and feel comfortable that the functionality is working as expected.

Types of Testing

Upgrades

Testing during an upgrade is a three-step process incorporating unit, system and user acceptance testing, as indicated above. These phases are outlined in the upgrade project plan and are critical to a successful go-live. Typically, each phase is scheduled for three to four weeks and will follow the testing completed by CMS Central. All processes in all modules must be thoroughly tested during an upgrade testing cycle. In addition, the conversion process is being tested to ensure data moved from one version to next successfully.

Major Releases

As baseline requires campuses to schedule releases to go into production several months in advance, testing can easily be scheduled to occur as soon as the release is available for the campuses and is put into a testing environment. Typically, there will be a minimum of two weeks to test a large release.

Functional leads will be responsible for reviewing all release notes on major releases and should ensure that their identified testers have also read the sections of the release notes that pertain to their testing areas. The CMS team will coordinate testing by notifying the leads when the releases are applied in a test environment and when they will be applied to production.

Updates, Fixes, Patches and Bundles

Updates, fixes, patches and bundles (updates) may be released by baseline outside of a major release, and will also require testing. There is typically one week available to test these items before they are put into the production environment.

Functional leads will be responsible for reviewing these items (typically done in the Instance Management meeting). The testing of these items may only require representatives from a few areas, but all areas should be aware of items that are going into production. The notification of when these items will be applied to test and production instances will be done via the cmstech ListServ prior to and following the Instance Management meeting.

Test Scripts

The CMS team, in conjunction with the functional leads, will coordinate test scripts for all functional areas (many have been developed by baseline; some will be developed on campus). These scripts will be used during each testing period, and will be related as appropriate to the various help desk tickets and other problems being fixed with the releases and updates.

Some scripts will be used during every testing period to ensure that all customizations are working properly and to ensure that problem areas are not impacted. Some scripts will only be used when their specific areas are impacted. Once the identified test scripts are completed, testers may test on their own as they desire.

In addition to individual test scripts, a summary testing spreadsheet will be provided to assist users in organizing their testing. This spreadsheet will include an area for the tester to sign-off on and indicate whether or not the test passed or failed.

What to Test

During each testing phase, it is important to understand what to test. For an upgrade, you must test everything: data entry and maintenance from an administrative and self service perspective, running processes and reports, interfaces and queries. When testing for major releases, you should test all these things as well, but focus on those items that have changed.

You should always test self service and any local customizations to ensure that the delivered changes in functionality have not impacted them. When testing for updates, fixes and patches, the primary thing to test is what is being fixed, but again, self service and local customizations should always be tested.

Reporting Problems

When problems occur during testing, they should be documented on the test script and logged via the CMS Help Desk for resolution. This will allow us to track the problems and their resolutions as we move through our testing phases. Issues with security should also be reported via the Help Desk where tickets will be logged to the CMS security coordinator and/or the CMS security administration team.