



JPL

Automated Software Testing for the Common Operating Environment

Eugene Jones
Eugene.Jones@jpl.nasa.gov
(626)351-1538

Distributed Systems Technologies Group
Section 369 – Mission Software Systems
Information Technologies & Software Systems Division



Introduction



- Problem: Too much code over too many platforms with too few resources to test every feature.
 - Common Operating Environment (COE) works on 33 different platforms of 5 different Operating Environments
 - Approximately 178,000 lines of code
 - 40% in C, C++
 - 40% in Java
 - 20% in scripts
 - Combination of Command Line Interfaces (CLI), Application Program Interface (API) and Graphical User Interfaces (GUI)
 - Software changes include correction of defects as well as introduction of new functionality
 - Both must be integrated into existing code baseline
 - Regression testing necessary to confirm that current capabilities are not compromised



Supported Configurations



Solaris 7		32 Bit	Single	Ultra	
		64 Bit	Multiple	Sparc	
			Single	Ultra	
Solaris 8	O/S Version 701	32 Bit	Single	Ultra	
		64 Bit	Multiple	Sparc	
			Single	Ultra	
	O/S Version 202	32 Bit	Single	Ultra	
		64 Bit	Multiple	Sparc	
			Single	Ultra	
Windows 2K	Server	Single Processor	Professional	Single	PIII
			Domain Controller	P4	
		Multiple Processor	Standalone	P4	
			Domain Controller	P4	
	Advanced Server	Single Processor	Domain Controller	P4	
			Standalone	P4	
		Multiple Processor	Domain Controller	P4	
			Standalone	P4	
Windows NT	Workstation	Single Processor		P4	
				P4	
	Server	Single Processor	PDC	P4	
			Standalone	P4	
			Standalone	P4	
		Multiple Processor	PDC	P4	
			Standalone	P4	
			Standalone	P4	
HP/UX-11.0		32 Bit	Single	B Class	
		64 Bit	Single	B Class	
			Multiple	L Class	



Background



-
- Task is to provide a “common environment” for users/administrators across heterogeneous platforms within a site/administrative domain.
 - Provide single point for user/group administration
 - Manage processes used/shared by different mission applications
 - Provide a baseline security configuration for each platform
 - Load software components onto each platform such that system conflicts are mitigated
 - Work is being done for the Defense Information Systems Agency.



Solution



- Partial Solution:
 - Use automated testing to supplement the existing testing staff
 - Provide better test coverage of new functionality
 - Provide better regression test coverage
 - Goal
 - Verify the nightly build
 - Regression test core GUI and application functionality.
 - Provide initial feedback before testing hours are expended.
 - Develop a process to compliment manual testing
 - Caveats
 - Automated testing assumed to be at most a partial solution
 - Automated testing tool limitations may restrict the types of tests performed
 - Testing must be completed by 8 am to be of use to the test staff.



Implementation



- Tool selected
 - Rational Robot version 2002.05.20 (selected to coordinate with customer testing)
- Configurations supported
 - Windows NT Workstation
 - Windows 2000 Professional
- Nightly Procedure
 - Manual Process
 - Restore the OS using Drive Image for test machines to known configuration
 - Configure tool which includes importing scripts and setting playback options
 - Set scan for installation flag sent from the build machine
 - Automated nightly build occurs for UNIX and Windows
 - Tool installs new Windows software build
 - Tool performs testing
 - Tests graphical interface
 - Tests for positive results and error conditions
 - Performs OS reboots
 - Modularized scripts for reuse



List of Tests Performed

JPL

- Software Installation
- Merge account information between two machines
- Reload the account management server
- Create user
- Create group
- Create profile
- Assign a password to a user
- Change to profile selector configuration
- Assume multiple profiles
- User account modification
- System process verification
- Profile removal



Results



-
- Time Frame: 12 months
 - 3 bugs detected
 - 7 faulty builds detected
 - Hours Invested
 - Approximately 6 work weeks used to develop the original tests
 - Approximately 2 hours a week needed for oversight
 - Manually run testing requires two and a half hours
 - Automated testing requires one and a half hours
 - Even though few bugs or faulty builds were detected, the tool demonstrates that basic functions of the software are not broken in current builds.



Future



-
- Amplify the number of metrics
 - Time to recover from nightly build problems
 - Time and workforce required for regression testing
 - Regression test coverage
 - Incorporate UNIX complement
 - Script extension
 - Integrate Unit Tests



Lessons Learned



-
- Automatic testing can provide consistent test coverage with breadth and depth.
 - Automatic testing can include API's, CLI's and GUI's.
 - Automated testing can supplement but not replace manual testing.
 - Automated testing requires an initial and ongoing time investment.



Appendix A

Details of Tests



- System Administrative Testing
 - First the Installer is opened.
 - The test segment is selected to be installed and the Conflicts, Requires, and Release Notes buttons are tested.
 - The segment is installed and the Verification Log is checked.
 - The SampleSW segment is installed, home directory creation, CDS entry, and the segment name addition to the Segment Installer list of installed segments is checked.
 - The Segment Installer is closed.
 - The Edit APM Configuration Tool is opened and the APM Master is changed and submitted.
 - The APM Authentication GUI is opened and the Local APM Authentication key is set.



Appendix A continued

Details of Tests



- System Administrative Testing continued
 - The machines are merged.
 - The APM Client tool is opened and the host list is checked.
 - The TweakUI utility is opened and secman is used as the user to login automatically.
 - The registry is edited and the next script to be run upon login is entered.
 - The reboot script is initiated.
- APM User Testing
 - After reboot, secman user is logged in.
 - Close the COE User login conformation window.
 - Execute reload the APM Server.
 - Verify reload message.



Appendix A continued

Details of Tests



- APM User Testing continued
 - The Profile Selector Script is run. The script tests to see if the profile is assumed or not, and if the profile is not assumed it is logged as a failure and assumes the profile. If the profile is assumed, it de-assumes the profile and verifies the profile is de-assumed before reassuming the profile.
 - The APM Client Script is run. The script creates a local account, a group, and a profile.
 - The Assign Passwords Script is run. The script attempts to change a user's password using an incorrect administrative password, and a correct administrative password.
 - The Profile Selector Config Script is run. The script begins by verifying the single option is selected then selects the multiple option. The script then proceeds to add the test profile to the secman user and assume the profile. Then the script de-assumes the profile and removes the profile from the secman user. Finally, it returns the Profile Selector Configuration tool back to the single option.



Appendix A continued

Details of Tests



- APM User Testing continued
 - The TweakUI utility is opened and the Administrator is used as the user to login automatically.
 - The registry is edited and the next script to be run upon login is entered.
 - The reboot script is initiated.
- TestAbortTwo Segment Testing
 - After reboot, Administrator is logged in.
 - The first process of this cycle is the installation of the TestAbortTwo Segment. The script begins by verifying that the TestAbortTwo Segment is not installed. Then the segment is installed.
 - The registry is edited and the next script to be run upon login is entered.
 - If any of the TestAbortTwo segment processes log files exist, they are deleted.



Appendix A continued

Details of Tests



- TestAbortTwo Segment Testing continued
 - The TweakUI utility is opened and the secman is used as the user to login automatically.
 - The reboot script is initiated.
 - After reboot, secman user is logged in.
 - Upon login the RunOnce process log is verified.
 - The TAB profile is created and assigned to the secman user.
 - The multiple profile option is selected in the Profile Selector Configuration tool.
 - The TAB profile is assumed.
 - The Session process is verified.
 - The registry is edited and the next script to be run upon login is entered.
 - The reboot script is initiated.
 - Once the user is logged in the session, periodic, boot, and background processes are verified.



Appendix A continued

Details of Tests



- TestAbortTwo Segment Testing continued
 - The registry is edited and the next script to be run upon login is entered.
 - The reboot script is initiated.
 - After reboot, the secman user is logged in.
 - secman is logged in and the session process is verified.
 - The TAB profile is de-assumed and the session exit process is verified.