



Etautotest : A simple and robust test automation solution

Shilpi Sood
Cadence Design Systems
56 A&B&C, NEPZ, Noida
ssood@cadence.com
this_is_shilpi@yahoo.com

Abstract:

Etautotest is a simple and robust test automation solution based on makefiles and PERL, which is developed in house to match the diverse testing needs of the Cadence Encounter Test suite completely. Etautotest and its peripheral scripts enable easy and customizable creation, running and maintenance of test cases. It can quite accurately track the dynamically changing statistics with less manual efforts. This has lead to achieving better quality results with reduced turn around time.

Keywords:

test automation makefiles perl

1 Introduction

1.1 Problem

The R&D teams were using manual methods or distinct tools within themselves for handling their regressions, none that could cater to needs globally.

1.2 Motivation

The Product Validation team took up the charter for validating the product extensively using a single unified test mechanism that could be automated to give precise results in lesser time.

1.3 Challenges

- With over 160 different commands which have thousands of keywords and as much as 1000 varying statistics in some logs, automation had to be done with utmost care and precision.
- Working with a cross-geographical team and having their belief and hence , inputs for the tool, right from the planning stage.

1.4 Work Model

- Received continuous feedback and enhancements from R&D right from the start.
- Division of tasks amongst the PV team and regular tracking of the progress
- Best methodology chosen after internal discussions and the chosen implementation formalized through rigorous testing.
- Reviewed existing R&D methodologies and incorporated their best in ours

PureConferences

www.pureconferences.com

Pure Conference Solution Pvt Ltd., A-108B, Sector 58, NOIDA 201301, India; Tel: +91 120 4621080

Page 1 of 8

- Integration of existing R&D test suites in common and more structured organization and adding new test cases for missing functionality.
- Once ready, interactive sessions/demos arranged for R&D to give them hands on experience. User-friendly documentation was also created for use offline.

2 Functionality

2.1 How does it work

Etautotest when run from the command line, recursively traverses the directory structure underneath, runs 'make' on every test case and finally generates a consolidated report of the run. The same is done on multiple platforms in parallel using clearcase views or cvs spaces.

2.2 Directory Structure

2.2.1 *at the top level*

top level Makefile – contains definitions for all commands

bin – contains etautotest and its peripheral scripts

filter - contains the filter files of all levels

test generated files – contains test cases categorized under different functional heads

autotest.log – contains the complete details of the regression run

2.2.2 *at the test case level*

local Makefile – contains test case specific information

src- contains source files for the test case

golds generated files – contains verified results

testresults – contains the output from the current running

status.diff – contains the status diff, if any

log*.diff – created for commands showing log diffs, if any

log*.bak – created for commands showing tolerance diffs, if any

test.log – contains the complete details of the regression run for the test case

2.3 Creation of new test cases

Etautotest scripts can automatically create new test cases from the available set of inputs

- From non-etautotest scripts - any external ET script that can be run from the command line can be converted to the makefile format directly.

- From generated logfiles - if there are no scripts and only the output result files are available, a script can be used to generate corresponding makefiles.
- From existing test cases - by creating a replica in the vobs that can be appended and checked in.
- Create an empty template – for a new test case, create a makefile template that can be used to quickly fill in the desired values.

2.4 Modifying existing test cases

Similarly, modifications in existing test case can be done with minimal efforts

- Adding new keywords to all instances of existing command(s) temporarily or permanently: by adding the new change to an existing target in the top level makefile
- Changes that need to be made to specific test cases only
 - temporarily – use make -n to make the change on the command line itself
 - permanently – a perl utility takes the command, the change and the test case(s) and makes a permanent change.

Also, test cases ignored while their reported bugs were being fixed, are automatically included in the main run as and when the problem gets fixed.

2.5 Running test cases

Etautotest can be run as default for running every test case or has simple options and predefined targets for giving customized runs. Here are some examples -

2.5.1 *From top level*

etautotest => will run everything underneath

etautotest -p litmus => will just run test cases in the litmus folder

etautotest -p customer_parts => will run predefined targets

etautotest -f list => randomly chosen test cases specified in a file list

2.5.2 *at the test case level*

make => will the run test case completely

make clean => will clean the generated outputs

make verify => will just verify the test case

make report_statistics => will just run this command

make -n => will just list the steps



2.5.3 *Ignoring testcases*

etautotest -p ignore => will ignore all the test cases with this file

etautotest -p ignore.lnx86 => will ignore these test cases for linux and run as usual for other platforms

etautotest -p ignore.bug1213 => will ignore this test case till the fix is available

2.6 Robust Filter mechanism

With a huge number of statistics changing for every command, there exists a four-step filter mechanism to filter out the unwanted stuff and carefully monitor the required ones. It is based on -

- Regular expressions for ignoring single lines
- Delimiters for ignoring big chunks of text by specifying their start and end points.

These four levels are -

2.6.1 *First level*

This is the generic global filter to unconditionally ignore all parameters or values that are likely to be different but not of relevance. It is applicable to all commands. For eg – date/timestamps , machine name etc

2.6.2 *Second level*

This is the application level filter. It filters out noise that is specific to a command. For eg – coverage for cmnd1 and memory for cmnd2 etc

2.6.3 *Third level*

Provides tolerance values for parameters that are critical to be monitored. At the end of every test case, the tolerance script applies default tolerances such as, for a Key Value of 100-200 a Percentage Tolerance of 4% is applied.

2.6.4 *Fourth level*

To override the default tolerances, one can provide the tolerance parameter in the local makefile for that test case and that will take precedence over anything else. For eg – tolerance < argument>

2.7 Detailed results

Running etautotest gives output in autotest.log at the top level and test.log in every test case.

Autotest.log provides info such as -

- Contains pass/fail information for each test case.
- Regression run statistics – build, platform, user name, date, time etc

- Overall statistics - the total number of test cases, number of test cases failed, passed and ignored etc
- Specifies the reason for failure – It gives the list of failures and their cause of failure such as
 - failure due to a core
 - failure due to difference in status
 - failure due to differences in logfiles
 - failure due to tolerances
 - the test case itself didn't run etc

whereas test.log gives the complete details of the execution of that test case and what failed at which level.

2.8 Reporting mechanism

Reporting is done via an automatic mail mechanism that sends the regression results to all the concerned people giving them the above information. The mail also has individual we links that give further details about every failure

2.9 Debugging failures

At the top level the complete suite of failed test cases can be analyzed with ease by using scripts that categorize the failed data based on different buckets such as failures grouped on the basis of commands or failures grouped as per the error messages etc. This reduces the time spent in identifying and analyzing repetitive problems.

As the need be, one can visit the individual test cases and check the test.log, status logs, log diff files, tolerance diff files etc for complete analysis. One can also compare the current results of a test case with its historical runs and study the improvements.

2.10 Regoldening

There are scripts that after verification can automatically regolden the new results based on the input provided by the user such as platforms, commands, single or multiple test cases etc

3 Advantages

3.1 Ease of use

- Easy to convert test cases from other regression mechanisms to etautotest.
- Easy to make top level as well as test case level modifications.

- Automatic updation of golden data.
- Availability of structured, user-friendly documentation online.

3.2 Flexibility

- Customized running of test cases based on predefined personalized targets.
- Customized generation of reports for easier analysis.
- Provision to automatically ignore different test cases for different runs.
- Customized environment settings for different users for different runs.

3.3 Scalability

- Hassle-free addition of any number of new test cases, new commands, new keywords in the existing infrastructure.

3.4 Portability

Portable across

- multiple platforms – such as linux, solaris, aix, hp etc
- configuration management tools – such as clearcase, cvs
- different file systems – such as afs, nfs

3.5 Avoids redundancy

- Avoids duplication by defining things at the top level makefile and referencing them in every local Makefile.
- Defining variables at the top of every local Makefile and using them throughout the file.

3.6 Reliable results with reduced turn around time

- Quicker analysis has resulted in reduced man hours without compromising on the quality of results obtained.

3.7 Unified test mechanism

- Everything can be run under one roof. Along with, extensive command-line testing, etautotest includes
- GUI testing through prerecorded test cases



- Performance analysis by carefully monitoring the performance parameters over different releases and generating excel sheet and graphs
- It could easily incorporate some of the R&D's existing utilities as it is
- Can be used to run test cases parallely on grid matrix etc

4 Success Measure

The usage of etautotest has lead to the reduction in runtime and analysis from 4 days, 4 people(128 man-hours) for 3 platforms to 4 days, 2 people (64 man-hours)for 4 platforms, per release even with the increase in the size of the test suite.

The success of this tool can be attributed to the continuous feedback from R&D and the constant endeavor of the test team to make every need into a possibility through effective planning and teamwork.

Biography



Shilpi Sood

Member Technical Staff
Cadence Design Systems, Noida