# Overview of the Regression Test Tool for P

Our regression test tool testP.bat (referred to below as “the tool”) takes a text file with the list of test folders as the first parameter. The tool recursively scans folders from the list looking for a configuration file “testconfig.txt”. Once a configuration file is found, the tool checks the folder name for it (referred to below as “parent”), which must be one of the “Pc”, “Zing” or “Prt”. The pair

(parent, configuration file)

determines which of the P executables - pc.exe, zinger.exe or runtime - the tool will be testing. The configuration file supplies a path to the .p file and parameters of the relevant P executable.

Conceptually, a separate “test” for the regression tool is this pair (parent, configuration file), which includes all information needed to run the corresponding P executable.

For the tool to accept test folder list, the test folders should obey some rules which are enforced by the tool. A test folder structure can have as many subfolders as needed. For example, a user might want to structure tests by a language feature or their result – see, for example, folder structure of the regression test suite described below. However, at the bottom of the tree, there should be subfolders called “test folders”. A test folder is named after the test and has a subset of the three subfolders Pc, Zing and Prt:

…\test\

Pc --mandatory

Zing --optional

Prt --optional

Note that Pc folder is mandatory, since pc.exe compiler generates files used by zinger.exe and runtime.

The rules for the test folder list are as follows:

* each of the Pc, Zing and Prt folders should contain a single config file called “testconfig.txt”
* no other test folder can contain config file with name “testconfig.txt”
* Pc subfolder should be present in each test folder
* no subfolders are allowed under Pc, Zing or Prt folders

Violations of any of the three rules above is a fatal error for the tool.

# Folder structure for regression test suite

Pre-configured regression test suite is located under “Plang\Tst\RegressionTests” and has the following structure.

At the top level, called “Feature”, the tests are divided according to the P feature being tested, in the way that is specified in “TestingFramework.docx”. For example, four feature test subfolders have a number in the name which corresponds to the subsection numbers in the “P Features to Test” section of “TestingFramework.docx”. In addition, there are two more subfolders “Integration” and “Combined” – see explanation in “TestingFramework.docx”.

* Root
  + State Machine Level Declarations (subfolder called “Feature1SMLevelDecls”)
  + Statements (“Feature2Stmts”)
  + Expressions (“Feature3Exprs”)
  + Data types (“Feature4DataTypes”)
  + Integration Tests (“Integration”)
  + Combined Tests (“Combined”)
  + Liveness tests (“Liveness”)

St the next level down called “type of error” level, each “Feature” folder has the following three subfolders :

* Static Error: Static analysis reports an error (as listed in “P COMPILER” section of “TestingFramework.docx”)
* Dynamic Error: Zinger reports an error
* Correct: Zinger does not report an error

At the next level, test folders for individual tests described in the previous section are located.

For example, here’s a full path to the test “function” that tests static error in the state machine level declarations feature #1:  
\plang\Tst\RegressionTests\Feature1SMLevelDecls\StaticError\function

Each test folder contains \*.p file and up to three subfolders: Pc, Zing and Prt.

Pc, Zing and Prt folders are the “leaves” in the folder tree, and each of these contains “testconfig.txt” and the acceptor (“golden” output). Some subfolders Pc, Zing and Prt might not be present for some tests. For example, for the static error tests, the only subfolder would be Pc. If a test is only intended for testing Zing (Prt), then there would be Pc and Zing (Pc and Prt) subfolders in the test folder.

# Specs and folders related to regression testing Plang\Doc\TestDocs\

RegressionTestTool.docx: instruction on running the regression test tool and adding new tests (this doc)

TestingFramework.docx: list of P features to test with hyperlinks to the existing tests

Tests.xlsx: list of all tests with more information about the tests: features

tested, test result, etc.

Primary Regression test suite: Plang\Tst\RegressionTests

# Regression process for developers

1. Ideally, each developer should run regression on the primary regression test suite

“testP.bat RegressionTests.txt”

before pushing the changes.

1. If regression fails, the developer responsible for the change should figure out why it fails.

See section “How to analyze regression results” below for tips on how to do it for all the failed tests at once.

If the acceptors have to change to accommodate new valid outputs, the acceptors should be reset.

**How to reset acceptors:**

To reset multiple tests, use

“Plang\Tst> testP.bat <path to the text file with test folders> /reset

Note that it is not possible to reset parts of a test related to an individual P executable, since we do not allow folders …\test\Pc or …\test\Zing or …\test\Prt in the folder list provided to the regression tool. So, if only an acceptor to, say, Prt executable has to be reset, the parent test directory should still be provided:

…\test\

and acceptors to all three executables would be reset.

This way, we quarantee that Zinger and runtime executables would not be run on stale pc.exe-generated files.

1. Run testP.exe again and make sure that all the tests pass.
2. Push the changes AND new acceptors (and testconfig.txt, if changed).

# How to analyze regression results

“testP.bat” generates a summary file “failed-tests.txt” that includes all folders with failed tests.

To look at the differences between the acceptor file “acc\_0.txt” and the newly generated output file “check-ouput.log” for all failed tests, a batch file “display-diffs.bat” is generated by “testP.bat”. For now, it supports “kdiff3” tool which you can install from here:

<http://sourceforge.net/projects/kdiff3/files/>

The batch file “display-diffs.bat” assumes that the full path to kdiff3 is added to the environment variable %PATH%, for example, as follows:  
set PATH=%PATH%;C:\Program Files\KDiff3

# How to run test tool

Command line

“Plang\Tst> testP.bat [<path to the text file with test folders>] [execsToRun] [/reset]”

will build a debug drop of PLang and run tests against it.

“execsToRun” parameter can be one of the following strings:

runAll (default)

runPc

runZing

runPrt

To run the regression test suite, use file

Tst\RegressionTests.txt

Note that in the file with test folders, the names of the folders should either be absolute paths, or paths relative to Tst folder. In the latter case, the folder name should start with either “.\” or with the name (no backslash). For example:  
Correct folder names:

D:\PLanguage\plang\Tst\RegressionTests

.\RegressionTests

RegressionTests

Incorrect folder name:

\RegressionTests

The testing framework does the following:

1. Runs P executable(s) specified in the “execsToRun” parameter: pc.exe, zinger.exe or tester.exe (for runtime).
2. Concatenates all output written to streams, the return code of the exe, and some additional files defined in “testconfig.txt” into a single output file.
3. If no “/reset” flag: checks if there is some “acceptor” file that matches the output file exactly.

The test passes if the output is accepted, otherwise it fails.

With “/reset” flag: acceptor file “acc.txt” is updated.

If there are failing tests, the tool generates two files:

failed-tests.txt : a list of all tests that failed, to be used for running regression again

on the failing tests only (as the first parameter for testP.bat)

display-diffs.bat : a batch file to run in order to display differences in acceptors for all failing

tests

To create a test case X, you need to do the following:

1. Create a new folder X under Plang\Tst\RegressionTests\F\E, where F is the “Feature” subfolder and E is “error type” subfolder (StaticError\DynamicError\Correct).

There are the following “Feature” subfolders:

Feature1SMLevelDecls

Feature2Stmts

Feature3Exprs

Feature4DataTypes

Combined

Integration

Liveness

Zinger

For more information on specific features in each category, look at the TestingFramework.docx document, sections “P Features to Test”, “Integration Tests”, “Combined Tests”.

1. Create a file called “X\X.p”, which is your test case.

It is very helpful to have a comment in the beginning of myfile.p which briefly describes the purpose of the test.

1. Create one, two or three “executable” subfolders (depending on the test): Pc or Pc + Zing or Pc + Prt or Pc + Prt + Zing. Some examples of exe subfolders for different “error types”:

“error type” exe subfolders

StaticError Pc

DynamicError Pc + Prt + Zing

Correct Pc + Zing

Note that in case of a “Correct” test, the runtime program is non-terminating – hence no Prt folder should be present for such tests.

1. In each exesubfolder, create a file called “testconfig.txt”, which will define how to run your test case.

In each exel folder, standardized testconfig.txt files are used. There are about 9 different templates for testconfig.txt. The template applicable to a specific tool folder is completely determined by the location of a specific test T, in particular, by the error type (Correct, DynamicError and StaticError) and the executable that is being run in a specfic subfolder (Pc, Zc or Prt). Use existing testconfig.txt files from the tests withing the same category as the new test.

1. Run “testP.bat” with the text file containing the full path to the folder X with “/reset” flag. Make sure that the result is what you expected.
2. The tools will run in the corresponding exe subfolders, starting with pc.exe, then tester.exe, then zinger.exe. A test case will be executed as if the working folder is X.
3. Even if the exe being tested crashes, this will still be captured without crashing the regression tool.

The last two steps are desirable, but can be delegated to Ella:

1. Add the test case to Plang\Doc\TestDocs\Tests.xlsx.
2. Add a hyperlink to the test file to “\plang\Doc\TestDocs\TestingFramework.docx”, to the relevant subsection.

The contents of “testconfig.txt” is a sequence of directives of the form “directive: data”. The possible directives are:

arg: An arg to pass to pc.exe, runtime or zinger.exe; if more than one arg directive, then args are passed in order.

inc: A file that should be included as output for pc.exe, runtime or zinger.exe; can be more than one such directive.

del: A file to delete before running test case, can be more than one such directive.

dsc: A description of this test case, including the subsection number from TestingFramework.docx of the

specific feature tested, for example: “dsc: 1.7. Transition to undefined state: error”

Additional tips on using the tool:

* if you include a generated output file in the acceptor criterion, make sure you delete the generated file first with a “del: file” directive. Otherwise, a stale file could be hanging around in the folder and accidentally cause a bad test to pass
* make sure the acceptor files and testconfig.txt are checked in
* make sure outputs can be tuned to remove timestamps, time-to-completion, absolute paths. All of these are machine specific and will be captured by the testing tool. One someone else’s machine, these values may be different, cause a good output to fail to the test
* if you want to run regression only on a specific feature folder F or on a single test X, run

“testP.bat RegressionTests\F” or “testP.bat RegressionTests\F\E\X”

* the regression tool runs pc.exe (under Pc folder) only once for each test, and zinger and runtime re-use the files generated by pc.exe. That means that it is dangerous to run Zinger or Prt without first running Pc. So, when running regression on a particular test, use the test folder (where .p file is located) as a parameter of testP.bat (and not Zc or Prt subfolders).
* if you only want to run regression on a specific executable, use /runXX option when running testP.bat. For /runZing and /runPrt options, the tool will always run Pc compiler first.