TEST APPRENTICE

This is the preliminary documentation for the first experimental version of the Test Apprentice. The purpose of this tool is to help with testing. It is eventually intended to generate and execute tests (it would be an AI application). In its current state it is just learning by watching what other testers do. But it is useful in this state because it can repeat exactly what other testers have done before.

The Test Apprentice is easy to use, you just type in the commands that you would use for doing the test the first time, with only an occasional extra command. To repeat a test, you only need to specify a special Test Apprentice function giving it the name of the test to re-perform.

The Test Apprentice groups test steps (commands) into tests, and groups tests into test suites. A test is a group of highly related test steps which are used to test a (small) portion of the system and which can be run independently from other tests (e.g. NS Filing directory enumeration). A test suite is a group of related tests which can be run together for convenience (e.g. NS Filing).

The Test Apprentice has the following limitations:

1. It can only record and repeat commands to the (standard) Lisp Executive, it does not record mouse operations or commands typed into other windows. It also cannot in general record input which is non-commands typed to the Lisp Executive (e.g. responses to the FILES? function's questions), but some commands record responses on the history list, so they are recorded (e.g. responses to the COMPILE function's questions). The rule of thumb to use here is if a REDO would re-perform those inputs, then the Test Apprentice has recorded them.

2. The number of test steps recorded for a particular test is limited to the size of the history list (usually 100 entries).

The following are the commands and functions available in the Test Apprentice:

ST — Start Test. A Lisp Executive command which starts the Test Apprentice recording the test steps of a test. An implicit ST is performed after an ET command or an ADD-TO-TEST-SUITE function. An ST can be performed at any time to restart recording of a test (and forget any recording in progress).

ET — End Test. A Lisp Executive command which completes the recording of the test steps of a test. This must have the name of the test as its single parameter (e.g. "ET NSFiling-1"). Test names only have to be unique within a test suite.

ITS — Ignore Test Step. A Lisp Executive command which ignores test steps in the current test. Currently this will only ignore the previous step, but in the future it will ignore specified steps. This is useful for removing mistakes from a test or removing miscellaneous commands not directly related to the tests. If you need to delete anything other than the last test step, then you must edit the test suite manually or restart the test using ST (the former can only be done after an ET, while the latter can only be done before an ET).

ITR — Ignore Test Result. A Lisp Executive command which ignores test results from test steps in the current test. Currently this will only ignore results from the previous step, but in the future it will ignore results from specified steps. This is useful for ignoring intermediate results. The Test Apprentice will check to see if the results from re-running tests are EQUAL to the results obtained when the test was recorded. For example, one test step may have returned a window as a result, then the next merely see if the type of the result

is WINDOW. In this case two different instances of a window will not be EQUAL, so the result of the first step should be ignored. If this becomes a problem EQUALALL may be used in a future version of the Test Apprentice, but even this does not always work (e.g. it does not work on windows).

ADD-TO-TEST-SUITE An NLAMBDA function which adds a test to a test suite. Its one required argument is the name of the test suite. If the test suite did not exist before, then it is created. Test suites are just variables that contain a list in a specified format, they can be saved in files by the File Package command VARS (or UGLYVARS if data structures are used in test results or HORRIBLEVARS if there are circular pointers). This function must be called before any test steps are started. It sets up all future tests to be part of this test suite until changed by another ADD-TO-TEST-SUITE.

EXECUTE-TEST-SUITE A LAMBDA function which executes all tests in a test suite. Its one required argument is the test suite to execute. This returns a true (non-NIL) value iff all tests executed successfully. An error message is printed out for each test which fails.

EXECUTE-TEST A LAMBDA function which executes one specific test in a test suite. Its required arguments are the test suite and the name of the test in the test suite to execute. This returns a true (non-NIL) value iff the test executed successfully.

The following are recommendations on using the Test Apprentice effectively:

1. Don’t use absolute command number references, use relative ones (e.g. use "(VALUEOF -2)" not "(VALUEOF 36)" if you are on command number 38). This is so the test will run correctly when it is re-run (if it were re-run and the VALUEOF was done on command number 67, then you would get some unexpected command’s result instead of what you wanted).

2. Two different data structures (defined with DATATYPE, ARRAY, etc., such as windows) will never compare EQUAL, even if they contain the same values. This means these should never be used as (non-ignored) test step results. Use ways of looking at the contents of the data structures instead.

The following is an example of a normal session with the Test Apprentice for recording a test suite:

```
34_(ADD-TO-TEST-SUITE NSFiling)
NSFiling
35_ST
Start-of-test-block
36_<test 1 step 1>
<test 1 step 1 result>
37_<test 1 step 2>
<test 1 step 2 result>
...
40_ET NSFiling-1
End-of-test-block
41_ST
Start-of-test-block
42_<test 2 step 1>
<test 2 step 1 result>
43_<test 2 step 2>
<test 2 step 2 result>
...
65_ET NSFiling-2
```

```
End-of-test-block
66_(FILES?)
the variables: NSFiling...to be dumped.
want to say where the above go ? Yes
(variables)
NSFiling  File name: NSFiling
NIL
66_(MAKEFILE '{Erinyes}<Test>Tests>OS>NSFiling)
{Erinyes}<Test>Tests>OS>NSFiling.;1
```

The following is an example of a normal session with the Test Apprentice for repeating a test suite:

```
19_(EXECUTE-TEST-SUITE NSFiling)
Executing NSFiling-1
Executing NSFiling-2
T
```