## WHAT HAPPENS WHEN YOUR TEST PROCEDURES 'AGE'?

Have you ever been in the situation where you've run the same procedures several times and now they are no longer finding defects or they need changing to still be relevant?

Think about it. You've had a software delivery, preceded by weeks of effort in writing your test documents. You test the software and report the bugs. A bug fix release comes eventually, and the relevant test procedures are run again to check the fixes. Then … there is a wait for another release with extra functionality and possibly more bug fixes. The questions then arise. How much of the old test procedure is still relevant? Is it worth updating?

Here are a few factors that may help you decide on the right answer for you.

- Firstly, always speak with your developers as early as possible before the new release to find out what is changing and what new functionality is expected. This is no guarantee, as you will always get omissions because the developers don't think some changes will affect you, but will help you to scope the task.
- If the same test procedure is run under the same conditions, the same results should be obtained, unless something has changed. So, it can be a good idea to keep some of the "old" test procedures unchanged to run as regression tests.
- If your test procedures are turning up no new defects or they find defects that other test procedures have also uncovered, they are serving no useful purpose (other than as potential regression tests). These need to be either changed to make them effective again or cast aside.

KEEP THEM, CHANGE THEM OR CAST THEM ASIDE?

If the decision is made to keep the test procedures, what can be done to make them effective again? Consider the following:

- Varying the initial conditions (to try to take the procedures through a different path in the software).
    - To achieve this, you can run the test procedures in a different order. This will only help if you don't re-initialise between tests, to increase the chance of finding cumulative defects. For example, if the same test is run multiple times, in different circumstances, try doing these in a different order.
    - The initial setup may also be varied. For example, if a test is always done from a cold start, try a warm start or do a 'soft' restart. If you are doing a data test and the fields are always empty to start with, try doing the test at a point where several entries are already filled in and need to be modified.
- Cut down and combine tests. The technique of taking pieces from tests and recombining them in a different way can result in other defects being exposed, because a different set of conditions will apply. For example, if separate tests exist for data being entered, modified, deleted and used, or links are being created, used, modified and deleted, then subsets of these functions can be combined into one short test that can be repeated for different data. This allows the test to cross functional boundaries and can find more defects by checking the interaction between functions.
- Change the test data/data inputs to gain better coverage. For example, if high range values have been used in a numerical field, try using low range values or zero. If tests are run at a particular data rate, try the same test at a different data rate or with different parameters.

If the decision is made to cast the test procedures aside, you still need to make sure that all of the application's "key functionality" is covered either by other test procedures or are included in your regression test suite.

  

TIPS ON PREVENTING AGING

Writing short tests (20 steps or less) that allow the input data to be varied easily without a lot of maintenance overhead can reduce the aging problem. For example, if you are writing detailed tests procedures, it is better to have a test step read ”Enter valid data _____ in the Duration field” rather than ”Enter 20 in the Duration field”. Tables of data can be used with the tests written to utilise data from that table. Then, in order to change your data set, the table is the only item requiring maintenance. Short tests are easier to maintain if functionality changes.

You can also reduce the risk of test steps becoming obsolete or have procedures be less maintenance by writing tests that are not keystroke specific and therefore not dependent on an exact user interface. For example, a test procedure may read “On the Settings menu, select the Time field. Enter 2:00 and press OK.” This would become “Set the time to 2 minutes in the future”. *This advice is given with the proviso that you do not materially impact the repeatability of the test steps*.

CONCLUSION

Schedules and manpower often make test management decisions to let test procedures go hard to justify, particularly when a lot of time and budget have been invested in writing them. Testers should try to balance the need to test effectively against the budgetary concerns of the company/project. Recognising the signs of aging in your test procedures, and taking early action, will reduce the agony of dropping whole sets of tests and allow your test suite to remain effective over time.

  