AN AUTOMATED TESTING ANALYSIS

By Cordell Vail

ADVANTAGES OF AUTOMATING

1. Good way to input very complex data that has to be exact every time and the input has to be repeated several times.

2. Large amounts of data that has to be input several times but can have changes in the data that can be entered from a spread sheet. Changes to the data on the spread sheet do not require changes to the code. These data driven test scripts are easy to maintain.

3. Tests that have to be run more than 4 times can be run exactly the same way every time. That is very difficult to do with manual testing.
4. Automated test scripts can be written to cover broad variations of options that can then be run as Critical Path navigation every time something is changed in the application to see if fixing one thing broke another. This would be very time consuming for manual testing.

5. Outside, skilled personnel can be hired to write the test scripts and then current employees can learn to maintain and run the test scripts without having to obtain that level of programming skill. The skilled outside personnel can be brought back in to make major changes to the code, thus saving the cost of having to hire a full time employee to create the test scripts.

6. The robustness of automated tests can be a tremendous asset. Testing a wide range of repeated areas of the application can be tested by automated testing that could never be tested by manual testers because of time constraints.

7. Modular programming techniques can cut maintenance time to a minimum if done correctly. Thus changes in the application become manageable with automated test scripts.

8. Comment lines in the automated test scripts can become the documentation for the test script. They are easy to maintain and print out. These test scripts can also be used as manual test scripts if necessary later.

9. Most of the commercial automated testing tool will now run script from most any programming language. So libraries can be created to run routines that may already exist which have been or can be created by the developers on the team.

10. Once the automated test scripts have been run, reporting test results are normally automatically produced by the automated testing tool. This can be a great help in reporting to management and tracking defects by the test team.

11. Automated navigational test scripts are normally quite easy to write in a manner that most of the functional navigation through the application can be retested every time a change is made to the application. This would not be feasible in most situations with manual testing.

12. Automated test tools are very useful in comparing baseline reports to current reports. If the application produces a great number of reports, the old reports can be compared to the new reports to make sure all calculations are still working correctly as long as the same data is being used for both tests. For manual testers to accurately check the calculations on reports of more that a few pages each, is almost impossible.

13. Test automation is not just a Functional Testing tool. Automated testing tools for the unit testing or development part of the software life cycle are also very available and actually more refined than the Functional Testing automated testing tools. It is common knowledge that errors found in requirements gathering, code design or Unit Testing are normally 4or 5 times less expensive to fix than if found in Functional Testing, and as much as 20 to 50 times as expensive to fix if found in production. Automation can help find bugs further up the testing life cycle.

DISADVANTAGE OF AUTOMATING:

1. It costs from 5 to 10 times as much to write an automated test script as a manual test script because it has to be done either with a record and play back tool or written as programming code. Therefore if a test is not going to be run at least 4 times it is not worth automating at all.

2. If the tests are written on an application that is constantly changing, then the maintenance costs will normally outweigh any benefit gained from the automated tests.

3. Automated testing ROI is normally only gained on long term testing projects where the tests are used for regression tests.

4. Automated test scripts only test the application for consistency not accuracy. When the automated test scripts are written any defects that would be found by that test script are already found by writing the test script. Running automated test scripts can not be used to find new bugs in the software, with the exception of using them as regression tests to see if the software was broken by fixing something else.

5. Not everyone on the test team can normally learn the testing tool and write automated test scripts. It is a very specialized skill.

6. Normally automated testing takes up more resources during the start up phase and when you are an environment with rapid release schedules, automated testing normally will slow you down over all.

7. Most main stream testing tools are very expensive to purchase and normally have a 25% annual maintenance fee. So if you are going to pay as much as $50,000 for a tool and $10,000 per year maintenance, you have to be doing a lot of automated testing to gain any ROI on that tool. Even Open Source automated testing tools are not free. Thousands of dollars are invested in time programming, maintaining and running those tools as well, compared to manual test scripts.

8. Making major changes in your software can make all the automated test script unusable. This is especially true if you have not used modular programming methodology for the automated test scripts (i.e. call statements).

9. When management makes a large investment in a testing tool, they normally expect a fast ROI on that investment. That simply will not be the case with an automated testing tool. That can be a hard sell for the test team to justify with management when months roll by with no results visible to them.

10. Normally automated testing requires more workstations to run. In a typical scenario an automated test engineer would have at least 2 computers and may have as many as 10 to 15 to run the test scripts all at the same time. If those computers are sitting idle much of the time when automated test scripts are not being run, that is not a good use of resources.

11. 100% test automation is not a realistic goal. Depending on the application sometimes not more that 10 % to 20 % of the application can be automated successfully. Often management expectations are much higher than that and those expectations can cause unrealistic pressure for results from the testing team.

12. To be a successful automated test engineer you must also have developer skills. If you can not program you likely will not be a successful automated tester. Most testers do not have programming skills. Most developers do not like to be testers. It is not easy to find a highly skilled developer / tester skill set in the same person.

13. In some cases automated test tool upgrades will make the old code unusable unless converted. In converting that code, it is possible to also cause problems with the interface to the application under test.

14. Automated testing can become very challenging in a complex environment with many varying platforms, operating states, operating systems, or other such varying test environments.

15. Most of the commercial automated testing tools run a subset of a programming language that is unique and proprietary to that testing tool. This will add to the learning curve of the test staff using the tool.

16. It is actually possible to write automated test scripts that actually do not test anything. This is not an uncommon occurrence for those new to automated testing. You have to test the tests to make sure they are testing the requirements they were designed to test. This is especially true after major changes are made to the application under test.

17. Dependencies are perhaps the biggest potential source of problems for automated testing. The improper handling of dependencies can cause failures in automated test scripts.

18. Once a test is run there is little chance that running that same test again on that same application in an automated test script will find a new bug unless there has been a new bug introduced into the system by fixes to the application. When the same, already run manual tests are run again, there is a greater likelihood of finding new problems both new and old because of the variability of the way the tests are run by the manual testers.