



TAC Managing Automation Complexity

Overview

To understand the TAC architecture, it is necessary to understand the complexity of automation. The shortcomings of record / playback (R/P) methodology of automation are well documented. Many of the problems of R/P can be addressed by explicitly programming scripts. But programming is expensive and highly technical. Test groups often do not have the resources necessary to support such a programmatic automation solution.

This is a serious problem. We have a need to do more testing. We cannot hire more testers, and the ones we have are already over-extended. Leveraging tools - automation of the testing - becomes a necessity to get more testing done.

Unfortunately, the tools by themselves do not do testing. Someone has to program the tools to get the testing done. Testers often do not have the development experience needed to do the programming, nor do they have the time. Anyway you look at it, it is a vicious circle.

Many organizations try to handle the complexity by ignoring it. Trying to use R/P techniques are really just a way of ignoring the complexity. After all, the vendors say that the tools will work as designed: how can that be wrong? When the inevitable failure comes, it is blamed on the testers who somehow did not implement the tests correctly. The tools get shelved. The testers get blamed. The organization suffers.

Managing Complexity

Test & Automation Consulting LLC has a better idea. Instead of ignoring the complexity, manage it.

Consider the history of programming Windows. In the beginning, a developer who wanted to program in Windows had to know the programming language C. The development environment was very complicated; the data structures and application-programming interface (API) were arcane and complex. Very few developers were considered experts and they commanded very good wages.

As time went by, more tools were developed for programmers to use. C++, application frameworks, and better debuggers helped to make Windows programming easier. More Windows application became available as the skill level needed to develop for Windows dropped. Then a huge revolution occurred with the advent of Visual Basic and Delphi. All of a sudden, even programmers with limited skills could become serious Windows developers.



Innovative solutions to maximize your testing investment.

It is not that Windows has become less complex. Indeed, the range of technologies which comprise Windows has become even more staggering: DDE and OLE, IP and RPC, COM and DCom: the amount of code intricacy is absolutely staggering.

The stratification of complexity has been the answer. Complexity is handled on different levels by different classes of developers using different tools. Incredibly talented programmers take highly complex technologies and create new programming languages that allow programmers to work with those technologies *without having* to understand them. The natural result of this is a hierarchy of developers. The real technical wizards end up writing tools and languages to help those developers who are not as technically adept.

This is the theory behind part of the Test & Automation Consulting automation methodology. TAC recommends handling the programmatic complexity of automation inside the TAC Test – Automation Programming Interface (T-API) Infrastructure and application specific framework. This code is highly complex. In effect, the combination of the Infrastructure and the application specific framework combine to create a new programming meta-language. This is used by the scripter who calls on the provided functions inside TAC template scripts to perform the tasks that comprise each test case. This is why we call it the T-API.

Because most of the code base (T-API) already exists before starting on an automation project, the TAC client gets the benefit of a jump-start with robust, debugged, and highly available code.

Implemented correctly, a test script becomes a series of calls to the framework and infrastructure functions. Because the complexity is built into the functions, the script itself has very little complex programming in it. This allows the scripter to be able to create very robust tests without having much programming experience.

The TAC T-API Infrastructure and programming methodology has several ramifications for our clients. These are listed below.

- There has to be at least one developer associated with the test automation project. The test group of the client does not need to have such an individual on their team; Test & Automation Consulting can supply the needed skills to build the application specific framework and guide the scripters. If the client wishes to manage costs more closely, they can hire a developer who can be trained and mentored by TAC to work with T-API and replace the TAC automator when ready.
- The creation of scripts using T-API becomes a relatively low skill task. Anyone with a few programming classes or a year's programming experience can create scripts. Good scripters can come from the ranks of:
 - Interns



Innovative solutions to maximize your testing investment.

- College co-ops
- New hires into development

Of course, TAC can also supplement staff by supplying scripters already familiar with the TAC methodology.

- Testers can remain testers. Because the skill set for testing is often completely disjoint with that needed for developers, testers often have no desire to work with automation. This works well with the TAC methodology. Testers can plan and create manual test cases and those tests that would likely be valuable as automated scripts can then be automated by the scripters.