

Test Automation: The Promise vs. the Reality

For organizations testing numerous releases of an application throughout the year, it's a good bet that automation tools have been considered if not already purchased. After all, test automation is supposed to excel at quickly testing the ever-expanding regression baseline as it becomes exceedingly costly to test manually.

As IT professionals, we all do our best to test the new functionality; everyone's eyes are on it. However, in spite of our best efforts, regression testing gets short shrift.

Sometimes we get lucky and everything works. More often we aren't so lucky. Our new functionality works fine but our users discover that pieces of existing functionality are now broken. As IT professionals, we want to catch those problems before our clients do.

Does utilization of test automation tools really allow you to tame the wild regression beast? The answer is a resounding, "Yes, but..." Test automation will save time and provide a significant return on investment if the tools are implemented correctly. This is a big "IF" and many organizations find that implementing a productive test automation environment is fraught with challenges.

The Trouble with the Path of Least Resistance

It is easy to understand why many companies purchase test automaton tools and then find themselves disappointed with the results when we analyze the buying process and examine our initial expectations.

Tool vendors want to sell tools and who can blame them? It's how they stay in business. When we buy their toolset and plan to use it for many years, we want them to be successful and to be there for us in the future.

If tool vendors want to keep the sales cycle as short as possible, they must sell the prospect on the tool's ease-of-use, i.e., Record & Playback. Watch any automated tool demo and the sales engineer will walk you through hitting the record button, executing the manual test and then playing it back. It does not elicit the "oohs" and "ahhs" it once did, but many competent managers still get sucked into the fallacy that Record & Playback will give them the automated regression baseline they need. They want it developed quickly so they suspend disbelief to that end.

The fundamental problem with Record & Playback is that it's analogous to hard-coding values in application programs. Seasoned developers know that it's inefficient to hard-code values that may change often. So why would you want to "hard-code," via Record & Playback, in automation scripts? The truth is you don't, especially if you detest spending inordinate amounts of time maintaining these scripts for future application releases. Yet, spending huge amounts of maintenance time is exactly what many companies do when Record & Playback is used to automate regression tests. Eventually, managers start asking, "Is it really worth it?" To ensure automation delivers the value your organization expected when purchasing the tools, you must have a more robust strategy than just Record & Playback.

Spend Time Now or Spend Much More Time Later A.K.A. Pay Now, or Pay More Later

If Record & Playback is not an effective way to automate, why do so many organizations use it and what are alternative options? There are several reasons why a company might use Record & Playback, but most are variations of these four:

- Lack of knowledge of alternative approaches.
- Vendors relying heavily on the Record & Playback function in tool demonstrations.
- Managers rushing to create automation baselines. Using Record & Playback takes less time than developing a manageable automation framework.
- Managers tasking non-technical testers with creating the automation suite.

Number three (3) is deceiving as Record & Playback takes the longest time to maintain. If you compare test automation to any other application you will realize that only a small percentage of time is spent implementing it as opposed to maintaining it. Do you want the majority of blood, sweat and tears to occur during the relatively small duration of implementation, or during the many years the automation baseline is in use, providing you don't scrap it first because of the maintenance difficulty.

Additionally, number four (4) may make sense on the surface, but is also deceiving. Why shouldn't a manager task testers with implementing a testing tool? We'll address this question shortly, but first, let's take a look at all the possible approaches/frameworks one could use to implement test automation. We'll include the pros and cons

of each, including Record & Playback. Of course, there are hybrids and variations of the five approaches below, but we'll just deal with the main ones.

Record & Playback - An automation tool generates scripts by recording user actions. The generated scripts can be played back to reproduce the exact user actions.

Pro:

- Quick setup time
- Easy to learn

Con:

- Difficult to maintain
- Does not accommodate multiple datasets
- Does not accommodate dynamic data

Data-Driven - An automation tool generates scripts by recording user actions. The "hard-coded" data is removed from the scripts and placed in external repositories. The generated scripts can be played back using multiple datasets from the data repositories. This approach requires some understanding of programming concepts such as parameterization and looping.

Pro:

- Quick setup time
- Can use multiple datasets with one script
- Provides separation between the data and scripts

Con:

- Multiple scripts to maintain
- Does not accommodate logic branching

Modular – Groups of functionality or screens are turned into scripts. By combining modules or scripts, the user can form automated test cases. There is limited amount of recording in this approach. Also, this approach requires an intermediate understanding of programming concepts.

Pro:

- Provides separation between the data and scripts
- Delivers script reusability
- Provides one maintenance point for each functionality or screen

Con:

- Longer implementation time
- Implementation team must have programming background

Keyword - Automated test cases are created based on "Keywords." The keywords drive the navigation, data and validation. The scripts get generated during the execution of automated test cases.

Pro:

- Separation between script, data, and application under test
- Keywords can be portable to application
- Easy to maintain
- Non technical personnel can setup automated test cases after implementation

Con:

- Requires extensive programming background to implement
- Longer implementation time
- Detailed knowledge of automation tools and concepts a must

Database -Automated test cases are created based on database entries. The application's objects, navigation, and data are stored in a database. The automated test case queries the database to drive the automated test.

Pro:

- Easiest to maintain
- Can perform mass changes with query, one location to manage objects, navigation, and data

Con:

- Requires extensive programming background to implement
- Longer implementation time
- Require knowledge of database concepts
- Detailed knowledge of automation tools and concepts a must

Obviously, many of these approaches require skilled developers to help implement and that leads us back to the question of, “Why shouldn’t a manager task testers with implementing a testing tool?” Just as a business user can use an application, but not necessarily have the technical skill to customize it and write complex scripts to improve its usefulness, a tester with no development background may not be the best choice to implement several of the frameworks above.

To ensure success, test automation must be approached as a rigorous project that requires analysis, strategy formation, planning, framework development and manual to automated test case conversion.

One of the keys to success is separation of complex development tasks from the easier scripting tasks. Handpick developers within the organization as part of your test automation project team and

pair them with the higher level testers that will be responsible for executing the tool once its implemented. Future maintenance will fall to those individuals within the team that have the skills to keep the baseline running smoothly.

If your resources do not have the automation experience or the time to take on an automation project, you can also bring in a test automation specialty organization that can provide a turnkey solution and then train and mentor your team how to maintain and execute the baseline moving forward.

Points to Ponder

Test automation will deliver on its promise of reduced testing time and increased coverage for regression testing, but it has to be implemented correctly with a clear, strategic approach. Whether your company is exploring an automated tool purchase or attempting to get an implementation back on track, make sure you consider the following:

- Is there enough money in the budget to implement a structured, maintainable approach rather than just enough money to purchase the tool?
- Is there a clear, strategic implementation approach that will keep maintenance time and costs reasonable as the regression test baseline grows?
- Have you selected the correct automation framework for your needs?
- Does your team have the experience, skills and time to implement the automation effort correctly?

Having sound answers to these questions will remove much of the stress that often surrounds automation and will give your organization the resource it needs to ensure existing application functionality still works as expected release after release.