



Test Automation ROI

Dion Johnson
DiJohn Innovative Consulting, Inc.
dionjohnson@dijohn-ic.com
www.dijohn-ic.com

Abstract

Being a software test automation professional requires more than just developing automated test scripts. You must be able to justify and even "sell" the use of test automation to management. This is often done by identifying automation's potential return-on-investment (ROI), a ratio of benefits to costs.

The ROI formula is very normally very "simple". Costs are fairly easy to quantify, but the value of potential benefits is far more difficult to nail down. Traditional ROI approaches often oversimplify many elements of test automation leading to inaccurate results. These inaccuracies then lead to unrealistic expectations on the part of management for test automation, and this can lead to failure—personal, team, and sometimes project.

In this presentation, Dion Johnson provides a realistic approach to computing test automation ROI including specific benefit categories and calculations. In addition, he will share some helpful tips for increasing your organization's automation ROI.

Introduction

If you are in a position that calls for you to develop automate test scripts to validate an application under test, you may find that you are called upon to justify your job function on a regular basis. This is the case particularly if you are involved in functional test automation. Performance test automators may not experience this issue as much, and that's because the benefits of performance testing are very clear to management. They know that application performance is extremely important to users, and is one of the first things that people notice. Management also realizes that performance/load testing is not logistically feasible to even consider doing manually. There's little chance of a project attempting to gather 5000 people in room with a stopwatch to test the performance of an application! So in their minds, by automating the performance tests, they have successfully replaced or eliminated the need for 5000 testers! Wow! This is great! The down side to this, however is that this same line of thinking gets transferred to functional automation. Management believes that if they can completely replace manual performance testing with automated scripts then they can surely do the same with functional testing. So when they repeatedly request the status of automation and find that 100% test automation has not been met, you once again find yourself justifying your job function. Will this cycle ever end? Possibly not, this is why you need to be able to make a case for test automation at a moments notice by presenting the return-on-investment being achieved through the automation that is being performed.

The calculation for ROI is the "return" from an action divided by the cost of that action, where the "return" is equal to the cost subtracted from the gain:

$$\text{ROI} = \frac{\text{Gains} - \text{Investment Costs}}{\text{Investment Costs}}$$

There are several approaches for calculating ROI including:

- "Simple" ROI Calculation
- Efficiency ROI Calculation
- Risk Reduction ROI Calculation

Each is detailed in this document along with a sample usage calculation, plus pros and cons.

"Simple" ROI

The "Simple" ROI calculation focuses on the monetary savings achieved through automated test execution. This calculation is useful when project personnel want to see benefits of test automation in terms of reduced testing costs. The

investment cost includes such automation cost as hardware, software licenses, training, script development and maintenance, and script execution and analysis. The gain is set equal to the pre-automation cost of executing the same set of tests.

Sample Usage Calculation

Let's say we're working on a project with an application that has several test cycles that equates to a weekly build 6 months out of the year. In addition the project has the following manual and automation factors that will be used in calculating ROI:

General Factors

- 1500 test cases, 500 of which can be automated.
- Tester Hourly Rate – \$60 per hour

Manual Factors

- Manual Test Execution/Analysis Time (average per test) – 10 minutes

Automation

- Automated Tool and License Cost (5 licenses) – \$20,000
- Automated Tool Training Cost – \$5,000
- Automated Test Machine Cost (3 machines) – \$3000
- Automated Test Development/Debugging Time (average per test) – 60 minutes (1 hour)
- Automated Test Execution Time – 2 minute
- Automated Test Analysis Time – 4 hours for 500 tests
- Automated Test Maintenance Time– 8 hours per build

In order to calculate the ROI, we need to calculate the investment and the gains. The investment costs can be calculated by converting automation factors to dollar amounts. The automated tool and license cost, training cost and machine cost are straightforward, but the other factors will need to be processed.

- Automated Test Development Time can be converted to a dollar figure by multiplying the average *hourly* automation time per test (1 hour) by the number of tests (500), then by the Tester Hourly Rate (\$60). This equals \$30,000.
- Automated Test Execution Time doesn't need to be converted to a dollar figure in this example, because the tests will ideally run independently on one of the Automated Test Machines.
- The Automated Test Analysis Time can be converted to a dollar figure by multiplying the Test Analysis Time (4 hours per week given that there is a build once a week) by the timeframe being used for the ROI calculation (6 months or approximately 24 weeks), then by the Tester Hourly Rate (\$60). $4 \times 24 \times 60$ equals \$5,760.
- The Automated Test Maintenance Time can be converted to a dollar figure by multiplying the Maintenance Time (8 hours per week) by the timeframe being used for the ROI calculation (6 months or approximately 24 weeks), then by the Tester Hourly Rate (\$60). This equals \$11,520.

The total investment cost can now be calculated as $\$20,000 + \$5,000 + \$3,000 + \$30,000 + \$5,760 + \$11,520$, which equals \$75,280. That's a lot of money! But before you decide to eliminate test automation from your project, let's turn our attention to the gain. The gain can be calculated in terms of the Manual Test Execution/Analysis Time that will no longer exist once the set of tests has been automated. The Manual Execution/Analysis Time can be converted to a dollar figure by multiplying the Execution/Analysis Time (10 minutes or .17 hours) by the number of tests (500), then by the timeframe being used for the ROI calculation (6 months or approximately 24 weeks), and finally by the Tester Hourly Rate (\$60). Note, manual test development and maintenance are not considered because these activities must take place regardless of whether or not a test is automated. The gain is therefore $.17 \times 500 \times 24 \times \60 which equals \$122,400! Still think \$75,280 is too much?

Inserting the investment and gains into our formula:

| | | | | | |
|-------|--|---|---|---|-------|
| ROI = | $\frac{\text{Gains} - \text{Investment Costs}}{\text{Investment Costs}}$ | = | $\frac{\$122,400 - \$75,280}{\$75,280}$ | = | 62.6% |
|-------|--|---|---|---|-------|

The ROI is calculated at 62.6%. Note that over time this ROI percentage will increase, because the tool costs eventually get replaced in the calculation by tool support costs.

Pros and Cons

The advantage to using this “Simple” ROI calculation is that the project dollar figure makes it good for communication with upper level management, because money talks! There are several cons, however. One being that it is difficult to get tester’s hourly rates, if you’re not a higher level manager, and normally the ROI is calculated by a test engineer or lower level manager. Also, this calculation oversimplifies some things. It assumes that the automated tests completely replace their manual counterparts, but that may not always be the case. There may be some elements of the automated test that still need to be executed manually. In addition, some automated tests contain elements that wouldn’t normally be executed manually, and therefore provides expanded coverage that this ROI calculation ignores. For example, manual execution may only verify a sampling of 10 data values for a field that can hold 200 values, in an effort to save time. Given that these values may be easily automated, all 200 values may be added to the automated test. Finally, this calculation can be a little misleading, because it gives the impression that the project budget will decrease, but the project budget seldom decreases due to automation. Funds are usually redistributed for greater testing on other parts of the application.

Efficiency ROI

This calculation is based on the “Simple” ROI calculation, but only considers the time investment gains, for assessing testing efficiency, as opposed to looking at monetary values. This calculation is useful for projects that have had an automated tool for a long enough time that there isn’t much need to give much consideration to its cost in the ROI calculation. In addition, it may be better suited for calculation by test engineers, because the factors used are easily attainable.

Sample Usage Calculation

The project example from the “Simple” ROI calculation will be used for this sample calculation also. The main difference is that the investment and gains will need to be calculated considering the entire bed of functional tests, not just the ones that are automated.

The investment cost is derived from calculating the time investment required for automation development, execution and analysis of 500 tests, then and adding the time investment required for manually executing the remaining 1000 tests. Calculations need to be done in terms of days as opposed hours, because the automated tests are ideally operate in 24 hour days, while manual tests operate in 8 hour days. Since tests runs can often abruptly stop during overnight runs, however, it is usually a good practice to reduce the 24 hour day factor to a more conservative estimate of about 18 hours.

- Automated Test Development Time is calculated by multiplying the average *hourly* automation time per test (1 hour) by the number of tests (500), then dividing by 8 hours to convert the figure to days. This equals 62.5 days.
- Automated Test Execution Time must be calculated in this example, because time is instrumental in determining the test efficiency. This is calculated by multiplying the Automated Test Execution Time (2 min or .03 hours) by the number of tests per week (500), by the timeframe being used for the ROI calculation (6 months or approximately 24 weeks) then dividing by 18 hours to convert the figure to days. $.03 \times 500 \times 24 / 18$ equals 20 days. Note that this number will be reduced when tests are split up and executed on different machines, but for simplicity we will use the single machine calculation.
- The Automated Test Analysis Time can be calculated by multiplying the Test Analysis Time (4 hours per week given that there is a build once a week) by the timeframe being used for the ROI calculation (6 months or approximately 24 weeks), then dividing by 8 hours (since the analysis is still a manual effort) to convert the figure to days. This equals 12 days.
- The Automated Test Maintenance Time is calculated by multiplying the Maintenance Time (8 hours) by the timeframe being used for the ROI calculation (6 months or approximately 24 weeks), then dividing by 8 hours (since the maintenance is still a manual effort) to convert the figure to days. This equals 24 days.
- The Manual Execution Time is calculated by multiplying the Manual Test Execution Time (10 min or .17 hours) by the remaining manual tests (1000), then by the timeframe being used for the ROI calculation (6 months or approximately 24 weeks), then dividing by 8 to convert the figure to days. This equals 510 days. Note that this

number is reduced when tests are split up and executed by multiple testers, but for simplicity we will use the single tester calculation.

The total time investment can now be calculated as 62.5 + 20 + 12 + 24 + 510, which equals 604.5 days.

Now turning our attention to the gain, you'll find that it can be calculated in terms of the Manual Test Execution/Analysis Time. The Manual Execution/Analysis Time can be converted to a dollar figure by multiplying the Execution/Analysis Time (10 minutes or .17 hours) by the total number of tests (1500), then by the timeframe being used for the ROI calculation (6 months or approximately 24 weeks), then dividing by 8 hours to convert the figure to days. This equals 765 days. Note that this number is reduced when tests are split up and executed by multiple testers (which would have to be done in order to finish execution within a week), but for simplicity we will use the single tester calculation.

Inserting the Investment and Gains into our formula:

| |
|--|
| $\text{ROI} = \frac{\text{Gains} - \text{Investment Costs}}{\text{Investment Costs}} = \frac{765 - 604.5}{604.5} = 26.6\%$ |
|--|

The ROI is calculated at 26.6%.

Pros and Cons

As discussed in the "Simple" ROI section, project dollar figures are seldom reduced due to automation, so using the Efficiency calculation allows focus to be removed from potentially misleading dollar amounts. In addition, it provides a simple ROI formula that doesn't require sensitive information such as Testers Hourly Rates. This therefore makes it easier for test engineers and lower-level management to present benefits of test automation when asked to do so. Many of the other cons still exist, however. This calculation still maintains the assumption that the automated tests completely replace their manual counterparts. In addition, it assumes that full regression is done during every build even without test automation. This may not be the case however. Full regression may not be performed without test automation, so introducing test automation increases coverage and reduces project risks as opposed to increasing efficiency.

Risk Reduction ROI

The Risk Reduction ROI calculation seeks to address ROI concerns left by other calculations by looking at automation benefits independently of manual testing. Test automation saves time in test execution, which provides testing resources with more time for increased analysis, test design, development and execution of new tests. It also provides more time for adhoc and exploratory testing. This equates to increased coverage, which reduces the risk of production failures. By assessing the risk of not performing automation (relative to potential production failures), and calculating the cost to the project if the risk turns into a loss, this calculation addresses the ROI relative to an increased quality of testing. Given the need for tester's rates, risk analysis, and cost calculations this formula may be best suited for calculation by upper level management when there's a desire to see organizational test automation benefits.

Sample Usage Calculation

The project example from the "Simple" ROI Calculation will be used for this sample calculation also, and the investment costs are calculated the exact same way. The investment cost is therefore \$75,280.

The gain is calculated in terms of the amount of money that would be lost if a production defect was discovered in an untested area of the application. The loss may relate to production support and application fixes, and may also relate to lost revenue due to users abandoning the application or just not having the ability to perform the desired functions in the application. Let's assume that the potential loss is calculated at \$500,000.

Inserting the Investment and Gains into our formula:

| | | | | | |
|-------|--|---|---|---|--------|
| ROI = | $\frac{\text{Gains} - \text{Investment Costs}}{\text{Investment Costs}}$ | = | $\frac{\$500,000 - \$75,280}{\$75,280}$ | = | 564.2% |
|-------|--|---|---|---|--------|

The ROI is calculated at 564.2%.

Pros and Cons

The advantage to using this calculation is that it addresses some of the issues left by other calculations by eliminating the comparison between automated and manual test procedures, and dealing with the positive effect of increased test coverage. The Risk Reduction ROI is not without its own flaws, however. One is that it relies heavily on subjective information. Loss calculations are not absolutes, because no one knows for sure how much money *could* be lost. It could be much more than what's estimated, or it could be much less. In addition, it requires a high degree of risk analysis and loss calculations. In real world projects, it is often difficult to get anyone to identify and acknowledge risks, much less ranking risks, and calculating potential loss. One final disadvantage is that without the manual to automation comparison it still leaves the door open for management to ask, "why not just allocate more resources for manual testing instead of automated testing?"

Ways to Improve ROI

There are several common factors that affect each of the ROI calculations, so improvements (reductions) to any of those factors will certainly improve ROI, regardless of how it's calculated. These factors are as follows:

- Automated Test Development/Debugging Time (average per test)
- Automated Test Execution Time
- Automated Test Analysis Time
- Automated Test Maintenance Time

Automated Test Development/Debugging can be decreased by increasing the initial time investment in creating a framework and process in which the automated test development will be conducted. This will have a big eventual payoff, because it imposes standards that will provide for faster automation.

Automated Test Execution Time can be decreased by better exception handling routines. Failed test steps and failed tests drastically prolong test execution, because the automated test tool spends time looking for what doesn't exist, and spends extra time verifying that it really doesn't exist. In addition, automated test failures often compound upon one another, so one failure may cause another, although subsequent failures aren't related to defective application functionality.

Automated Test Analysis Time can be decreased by building better reporting mechanisms into the tests. This may be accomplished by adding special logging statements to the automated tests that will print to the report in the event of a failure. In addition, you may have the test trigger a screen capture in the event of a failure, so that it is easy to know what state the application was in at the time of failure.

Finally, Automated Test Maintenance may also be decreased by increasing the initial time investment in creating a framework and process in which the automated test development will be conducted. In addition, increasing modularity and adding comments to automated tests is another to make maintenance easier and faster.

Conclusion

There are several methods that may be used for calculating the automated testing return-on-investment. Which should be considered the "real", ROI? Which measure provides the most accurate view of the benefits of test automation? Which measure should we be using? The answer to these questions is: which ever one works best for your environment. The most important thing is for your organization to see the value in test automation so that it will continue to invest in it. In addition, which ever ROI calculation you use, you should be able to attain information that will help you to become more efficient in how your team conducts test automation. If possible, use multiple ROI calculations because the more calculations you use, the clearer your test automation efforts becomes.