

# The ROI of Test Automation

by Michael Kelly  
[www.MichaelDKelly.com](http://www.MichaelDKelly.com)

## Introduction

With the exception of my first project team out of college, in every project team since, I've had to explain either what automated testing is, why we need to do it, or what value it brings to the project. This can be difficult as each time I explain automation, I have to battle different preconceptions and experiences. Sometimes I even find myself having to develop a basic understanding of testing and software development. Wouldn't it be great if every project manager had to have a testing background or at least actual development experience? Probably not, I'm sure that would introduce all sorts of other problems. The point is it is a unique and difficult battle every time.

Educating people about automated testing requires time and mutual respect. Normally both of these are lacking on projects that are running late, are shot on budget, and have high visibility. Often automation is thought of as way to reduce the costs of testing, increase coverage, and shorten the testing cycles required for the project.

This paper looks at the different types of test automation offers guidelines for how to calculate their return on investment. The topic of test automation is a large one and because every context for which automation is implemented is different, I cannot offer bulletproof equations that will work for every project. What I can offer are some general guidelines, some ideas for maximizing your return on investment, and one or two examples of ROI calculations that are commonly used. The purpose of this article is to better enable you to communicate the value of automation and determine what types of automation will yield the greatest return in your specific context.

## Why you need ROI analysis

In general, automated testing involves higher upfront costs for a project while providing reduced execution costs down the road. Performing return on investment (ROI) analysis on each automation project will determine a simple approximation of cost, will help determine upfront what types of automation you want for the project, what tools will be required, and what level of skill will be required for the testing resources for the project. Not only does ROI serve as a justification for effort, but also a necessary piece of the planning process for the project.

It has been my experience that projects that do not perform ROI calculations upfront do not fully understand the costs of their automation effort, what types of automation they could be doing vs. what they are doing, and what strategies to follow to maximize their return. The projects I've worked on where the ROI for the effort was looked at up front had better and more hardware and software resources (due to justifications). They also had more focus for test planning and test case/script development, and achieved more goals than those that did not.

## The classic example for calculating ROI for automation

Without wasting more time, let's look at the classic example for how to calculate the ROI for an automation effort. This classic example (which is a combination of the examples found in the articles by Varghese and Hoffman referenced below) for ROI is calculated by dividing the net benefits of the automation effort by the cost of the automation effort, or

$$ROI = \frac{\text{Benefit}}{\text{Cost}}$$

Traditionally the costs used are the costs for automation (hardware, software and licenses, training, time to produce scripts, etc...) The benefits are calculated by looking out over a time interval at the savings automated testing offers over manual testing (reduced time to execute tests, ability to test 24 hrs a day, etc...).

Using that model, we could calculate the ROI for a given project the following way. Let's look at the automation of testing for a small website that gets weekly content updates. We would start by looking at the costs for the effort in terms of time.

Item	Cost	Time
Publish website and execute testing		Weekly
Develop manual tests for the site	1 tester @ \$50 hr	4 days
Execute manual tests for the site	2 testers @ \$50 hr	1 day
Develop automated tests for the site	1 tester @ \$50 hr	10 days
Execute automated tests for the site	1 tester @ \$50 hr	1 hour to review logs
Maintain manual test cases	1 tester @ \$50 hr	1 day every two weeks
Maintain automated test scripts	1 tester @ \$50 hr	1 day every week
Hardware for test execution	2 computers @ \$1,000 ea	
Test software and licenses	2 licenses @ \$2,000 ea	

If we know the timeline for the project, let's say we knew this website would be updated weekly for the next year, we can calculate the return on investment for automation.

*Cost of automation* = price of hardware + price of software + time to develop scripts + (time to maintain scripts x number of times scripts are executed) + (time to execute scripts x number of times scripts are executed)

*Cost of automation* = 2 computers @ \$1,000 ea + 2 licenses @ \$2,000 ea + 10 days @ \$50 hr + (1 day @ \$50 hr x 52 weeks) + (1 hr @ \$50 x 52 weeks)

*Cost of automation* = \$2,000 + \$4,000 + \$4,000 + \$20,800 + \$2,600

*Cost of automation* = \$33,400

*Cost of manual* = time to develop test cases + (time to maintain test cases x number of times tests are executed) + (time to execute manual testing x number of times tests are executed)

*Cost of manual* = 4 days @ \$50 hr + (1 day @ \$50 x 26 weeks) + (1 day @ \$100 x 52 weeks)

*Cost of manual* = \$1,600 + \$10,400 + \$41,600

*Cost of manual* = \$53,600

*ROI* = benefit / cost

*ROI* = (cost of manual – cost of automation) / cost of automation

*ROI* = (\$53,600 - \$33,400) / \$33,400

*ROI* = about 61% over the course of the year

### What's wrong with this?

There are a couple of problems or incorrect assumptions with this model. First, you can't compare automated testing to manual testing. They are not the same and do not provide the same information about the application-under-test. Automated tests are by their very nature simpler than manual tests. Automated testing is simply not as effective as manual testing is for complex tests. Because it has to be repeatable, must work within the confines of the tools and scripting languages used, and must have some semblance of maintainability, automation lends itself to certain classes of tests (discussed later in this paper). Further, aside from types of tests, automation will find different types of bugs than manual testing

will. You will discover bugs with automated tests that may be impossible to discover with manual testing. Performance testing or generating extremely large sets of test data are good examples of this.

The second problem is that you can't compare the costs of multiple executions of automated tests to one execution cycle of manual tests. Because the types of tests are different your automated tests can't perform all of the types of testing you would perform manually, and you would never dream of executing many of your automated tests manually. Imagine paying someone to sit in front of a computer and check off all three thousand label names for the application? It's completely silly. While this is a good automated test, it would never be done manually.

When calculating ROI for automation you need to look at the real value automation is giving you. This is not to say the above equation will *never* work. It will work for some extremely small subset of the automated tests and manual tests you create, but it is certainly not representative of standard regression test automation or the various other types of automation you could potentially be scripting. What we really need to look at are the individual returns for automation, independent of the manual effort. Most likely, you will develop a plan utilizing both methods of testing, and you will be evaluating each type in terms of its own independent worth.

### **Looking at the real ROI**

The real return on investment for automation is based on each type of automation you choose and what value it adds to the overall testing effort. Think of performance testing. When is the last time you had to justify performance testing to your boss? My bet is that in the last few years, never. People realize that slow websites will lose customers and that server crashes cost money. You don't need to explain that to them. They may not know anything about performance time requirements or user distributions, but they know that performance testing needs to be done. The value of performance testing is readily apparent and is not based on the costs of automating. Moreover, automation is the only practical way to perform these types of tests.

Similarly, each type of automation will have its own unique return and that return may or may not relate to a manual testing activity. It will be made up of both tangible benefits (automated data generation will save manual testers 15 hrs) and intangible benefits (by developing automated scripts as a parallel effort to screen development testers are seeing the product sooner, helping developers with their unit testing, and can catch bugs sooner in the software lifecycle). When looking at intangible benefits, we know that the activity is good and has value, but how much money does it save? There is often no formula available to calculate the savings.

This makes calculating the ROI an exercise in identifying those types of tests that are required for the project and thus don't need to be justified and then performing cost benefit analysis on the other types of automation being considered to see if they are worth the cost and effort. Once all the candidates for automation have been identified, their values have been weighed, and the costs have been determined, then ROI for each type of testing needs to be calculated. From there you can justify each cost to management, and you can plan the entire testing effort accordingly.

### **Don't forget your context**

Each project will require different types of automation, will assign different tangible and intangible benefits to automation, and will have different types of required automated tests. Virtually all web applications can relate to the performance testing example I've used, and I'm sure most real-time systems and resource intensive applications like games and CAD tools also do performance testing as a matter of principle, but certainly, performance testing may not be a hard requirement for applications such as calculators, text editors or other small desktop applications. While there are definitely arguments for why you would want performance testing on those types of applications, there could be good arguments for why the cost is not justified.

What context does your application exist in and what types of requirements are you testing? What are the skills of your staff? What tools do you have available? How much time do you have for testing? How

much time do you have for training? All of these questions (and certainly more) need to be answered before you can start looking at the real ROI for any given type of automaton for your project.

### **What are you automating?**

There are many different types of test automation. One of the problems with the classic example of ROI for automation is that it only looks at one type of automation, regression scripts. There are however many types of test automation. Test automation can be as simple as automating the creation of some of your data, or as complex as a series of scripts/programs that do parallel testing of two or more systems with live data. A small sampling of all of the types of automation available is listed below:

- Load and performance testing
- Installation and configuration testing
- Testing for race conditions
- Endurance testing
- Helping the development effort with smoke tests and unit tests
- Analyzing code coverage and runtime analysis
- Automation of test input generation
- Checking for coding standards and compliance
- Regression testing
- Many more...

Your goals for automation will also affect what you automate. Do you want to find bugs? Do you want to establish tractability for some sort of compliance? Do you want to support the development team? Do you want to establish scripts that allow you to ensure nothing changes in the software? These questions are a small sampling of all of the questions that need to be asked in order to determine the scope of your automated testing. Related to your goals, your software development lifecycle and the skills of your testing staff will also affect what you choose to automate. Are you developing in an XP lifecycle with programmers also testing? This would probably lead to more automation. Are all of your testers junior? You probably will have very little automated testing or automated tests that are shallow.

Your automation effort will also depend on the tools you use and have access to. If you already have an investment in an enterprise test tool, you will probably want to leverage it. If you are operating on a tight budget, you will look more at open source tools, shareware, and custom tool development and scripting. Some tools are designed for regression testing (typically these are the GUI test tools), some for performance testing, and many are designed for specific types of testing or for specific environments or technologies. As with the types of testing, below is a sampling of the types of automation tools available:

- Any scripting language
- Disk imaging tools
- File scanners
- Macro tools
- Memory monitors
- Environmental debuggers
- Requirements verifiers
- Test procedure generators
- Syntax checkers/debuggers
- Runtime error catchers
- Source code testing tools
- Static and dynamic analyzers
- Unit test tools
- Code coverage tools
- Test data generators
- File comparison utilities
- Simulation tools
- Load/Performance testing tools
- Network testing tools
- Test management tools
- GUI testing tools
- Environment testing tools

### **What will that cost?**

So how do you calculate the costs for all of these types of tests? It's not too dissimilar to the classic example shown above. You still need to look at all the fixed costs for each test. The only thing that has changed is that individual cost needs to be determined in a slightly different way. Because each test needs to be judged based on its own merits, you will want to look only at the costs specific to that type of automation. Most likely, that means specific software or a specific skill set.

What you need to avoid is looking at the costs that would be present for any type of automation. That is, don't look at the hardware costs unless they exceed the hardware used for everyday testing. You don't need to calculate the cost of computers for test data generation if all testers use computers regardless of whether or not you automate. However, you do need to include hardware costs if you need ten more computers to execute regression scripts on them. This method assumes that some of the costs for automation are simply absorbed by the testing group as costs that would exist regardless of automation. Testers will still have computers. Testers will still need training. And so on.

### **What are the benefits?**

Calculating the benefits for each type of automation involves an in-depth look at each type as it exists in your specific context. For example, it would not be unreasonable for the automation effort to assist in the unit testing of screens as developers start churning them out. Some potential benefits of this are time savings for developers (they are often more critical path for the project and there may be a money savings if they are paid more per hour). This also finds more bugs earlier in the process, which means they get fixed before the initial release to testing. This also saves money and could be calculated by looking at the average cost to fix bugs on similar projects. Finally, this also gets the tester working with the screens earlier rather than later. This means the tests developed could be used as a baseline for other automation efforts (this value can be determined and used for other automation ROI calculations), and it means testers will have a basic understanding of the code before the official test cycle begins (also saving money as they will not have to spend as much time familiarizing themselves with the software).

Another example could be automating installation testing. Let's say we need to test the installation of a program on four different operating systems on ten different hardware configurations. This would be 40 different tests, all very similar, none of them overly interesting for the tester, and all of them requiring significant time as far as the time to install and performing the verification of a "correct" install. Here we could perform the test one time manually, and use those numbers to determine the time to install the software, determine the time to automate the install of the software, determine the cost of developing the validation scripts, determine the cost for manually executing testing on each configuration, etc.... All said and done, we can use equations similar to the classic example above to determine if this automation makes sense. We can also factor in if this testing needs to be executed for each iteration.

The real determining factor here is the level of detail in which the validation scripts will go. Ideally, the validation scripts will not be the only test you have for all these divergent configurations. The scripts will most likely fulfill your testing for a specific subset of requirements. Again, this is not a substitution for manual testing but is its own breed of testing. The ROI value is not the value of automation vs. the cost of executing these tests manually, but is instead the benefit of this type of testing, plus the benefit of whatever the manual tester is doing while these simple tests are executing.

### **Calculating ROI**

A general rule I follow for automation is to automate all things that offer immediate returns and to automate the tests I couldn't possibly perform manually. Unfortunately, as pointed out above, these tests normally don't need to be justified anyway. Nevertheless, don't overlook them as legitimate types of automation and deduct their costs from the costs of other types of automation. If you have powerful hardware and software for performance testing, there is a good chance those resources will be useful for other types of testing as well.

Rule number two is if you have a hard time convincing someone that it needs to be automated and if you feel you need an equation to see the long term advantages of automation, don't bother automating it. As proven by modern accounting scandals, you can make the numbers say whatever you want. Don't use ROI calculations as confirmation for a bad idea. If the value of automation is not apparent, these calculations will not make them apparent. These numbers will only help with the planning of automation and serve as a quick justification for resources. Don't use them to fool yourself into automation that you don't need.

Finally, when you have to, fall back to the classic example above for how to make the calculation. Try not to equate the effort with manual testing as a means for cost cutting, but instead look at what it does in terms of freeing up your manual testers. In the installation and configuration testing example above, I freed up a manual tester by automating that obviously boring and repetitive type of testing. This does not mean I'm not paying that tester anymore. It means she is doing more challenging and more meaningful testing, like helping that junior java developer track down a potential memory leak.

### **More information**

For more information on calculating and maximizing the ROI for test automation reference the following sources:

- *Lessons Learned in Software Testing* by Cem Kaner, James Bach, and Bret Pettichord
- *Cost Benefits Analysis of Test Automation* by Douglas Hoffman
- *Test Automation – An ROI based Approach* by Jose Varghese
- *Maximizing ROI and Avoiding the Pitfalls of Test Automation* by Bill Hayduk