# Simple ROI model for Testing Automation projects

**The writer is an independent consultant for automation testing in the fields of networking and J2EE applications. Guy_Arieli@hotmail.com. Tel: +972-54-7899446.**

## *Abstract*

In this article I will try to give a simple method to answer a question that usually rises when working on testing automation project, “Will it be profitable”? A more accurate question will be “When will I See the return on the investment”?

In general Return On Investment or ROI is a factor that is calculated in specifics points in time. When the ROI become positive the project worth investing.

Most of the ROI models are very hard to implement. Here I will give a model that by answering few simple questions will give you a good first appraisal.

I found this model to be a strong tool to priorities between automation projects (What should I automate first?) and it can help you understand what are the factors influencing your project.

## *The model*

The units I use to measure costs are working weeks. The calculation is done for every major release of your product.

The $ROI$ index is the Total Benefits minus the Total Costs divided by the Total Costs. So when the Benefits is bigger then the Costs you will get a positive number, and everyone is happy. The big question is “When?” $i$ is the product version index.

$$ROI_i = \frac{TB_i - TC_i}{TC_i}$$

Lets drill into the Total Costs ($TC_i$) first.

$$TC_i = FC + TBB_i + TTD_i$$

The Total Costs for version $i$ will be the costs to develop the automation framework ($FC$) this effort is invested once. To it we add the total cost to develop the building blocks of the tests ($TBB_i$), the interfaces to the system or product that is being tested. Plus the total costs to develop the tests ($TTD_i$).

Now lets look at the tests development:

$$TTD_i = (1 + TMF) \times TTD_{i-1} + TD_i$$

$TMF$ is the Tests Maintenance Factor. If it took you 4 months to develop the tests to the first version and you have to invest 1 month in order for them to run on the second version, then the $TMF$ will be 0.25.

So to calculate the total tests costs in version $i$, we take the total costs for the previous version ($i-1$) multiply it by 1 plus $TMF$ and add to it the efforts to develop new tests for the new version ($TD_i$)

We will do the same for tests building blocks:

$$TBB_i = (1 + BMF) \times TBB_{i-1} + BB_i$$

$TBB_i$ is the Tests Building Blocks Maintenance Factor.

$BB_i$ is the Building Blocks development efforts for version $i$. If for example they just add some kind of Database feature in the product and you have to write an easy to use API or a driver to enable access to the database, in order to test it. The development efforts will be added to the Building Blocks development ($BB_i$).

Now if we look at the 2 formulas (for $TTD_i$ and $TBB_i$) the only parts that are difficult to calculate are $TD_i$ and $BB_i$. $TBB_{i-1}$ and $TBB_{i-1}$ are 0 for the first version and are known for all the other versions.

So in order to make it easer to calculate $TD_i$ and $BB_i$ I use the following formulas:

$$BB_i = NF_i \times M_i \times BBM$$
$$TD_i = NF_i \times M_i \times TDM$$

Were:

$NF_i$ is the New Feature factor, the ratio between the efforts to test new feature and the total efforts (include regressions), of the manual testing. If half of the testing efforts are being invested in new features then $NF_i$ will 0.5.

$M_i$ is the manual efforts for version i.

$TDM$ is the ratio between tests development efforts to manual efforts. If I have a tests plan that take 1 week to execute and (assuming I have the tests building blocks) it will take me 2 weeks to write (and debug) the automatic tests the $TDM$ is 2.

$BBM$ is the ratio between building blocks effort ($BB_i$) to manual efforts.

We finished with the costs let's look at the benefits:

$$TB_i = TB_{i-1} \times RF_i + M_i$$

$RF_i$ is the Relevancy Factor, the ratio between the relevant efforts to the total efforts. If 10% of the tests I wrote to version 1 are irrelevant to version 2 then $RF_i$ will be 0.9.

So the total benefits for version $i$ equals to the benefits of version $i-1$ multiply the relevancy factor plus the manual efforts for version $i$.

I hope you didn't get lost in all the formulas. But in order to use this module on a project all you have to enter are the following parameters:

| Parameter | Description |
|---|---|
| $FC$ | Framework cost |
| $BMF$ | Building blocks maintenance factor. |
| $NF$ | New factor, the ratio between the new feature efforts and the total efforts (include regressions), of the manual testing. |
| $RF_i$ | Relevancy Factor, the ratio between the relevant efforts to the total efforts. |
| $BBM$ | The ratio between building blocks effort ( $BB_i$) to manual efforts |
| $TMF$ | Tests maintenance factor. |
| $TDM$ | The ratio between test development efforts to manual efforts. |
| $M_i$ | Manual efforts for version i. Includes all the regressions and repetitions of tests. |

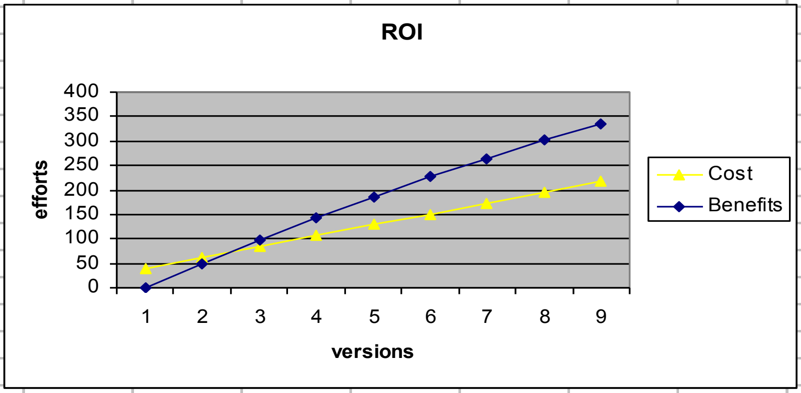
Only 8 parameters and you have it all.

## *Example*

Let’s see how it looks in excel sheet:

| | | |
|---|---|---|
| BMF | 0 | |
| TMF | 0 | |
| NF | 0.3 | |
| BBM | 0.5 | |
| TDM | 1 | |
| Mi | 50 | weeks |
| TC | 16 | weeks |
| RF | 0.95 | |

| Version | Mi | Nfi | Bbi | TBBi | Tdi | TTDi | Tci | Bi | ROI |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 50 | 0.3 | 7.5 | 7.5 | 15 | 15 | 38.5 | 0 | -1 |
| 2 | 50 | 0.3 | 7.5 | 15 | 15 | 30 | 61 | 50 | -0.18033 |
| 3 | 50 | 0.3 | 7.5 | 22.5 | 15 | 45 | 83.5 | 97.5 | 0.167665 |
| 4 | 50 | 0.3 | 7.5 | 30 | 15 | 60 | 106 | 142.625 | 0.345519 |
| 5 | 50 | 0.3 | 7.5 | 37.5 | 15 | 75 | 128.5 | 185.4938 | 0.443531 |
| 6 | 50 | 0.3 | 7.5 | 45 | 15 | 90 | 151 | 226.2191 | 0.498139 |
| 7 | 50 | 0.3 | 7.5 | 52.5 | 15 | 105 | 173.5 | 264.9081 | 0.526848 |
| 8 | 50 | 0.3 | 7.5 | 60 | 15 | 120 | 196 | 301.6627 | 0.539095 |
| 9 | 50 | 0.3 | 7.5 | 67.5 | 15 | 135 | 218.5 | 336.5796 | 0.54041 |
| 10 | 50 | 0.3 | 7.5 | 75 | 15 | 150 | 241 | 369.7506 | 0.534235 |



## Typical Projects

Following are list of project types with typical parameters values (for medium size projects):

| | $FC$ | $BMF$ | $TMF$ | $NF$ | $BBM$ | $TDM$ | $RF_i$ | $M_i$ |
|---|---|---|---|---|---|---|---|---|
| Typical networking project | 16 | 0 | 0 | 0.3 | 0.3 | 0.5 | 0.95 | 40 |
| Complex Dynamic HTML GUI | 16 | 0.3 | 0.2 | 0.3 | 0.7 | 0.7 | 0.9 | 40 |
| Functional testing for J2EE project using business logic layer (not GUI) | 16 | 0.1 | 0.1 | 0.3 | 0.2 | 0.5 | 0.95 | 40 |

## Limitations

- This model works best if you already have data of previous projects. It will enable you to calibrate the parameters values.
- The benefits include only direct benefits, the time that was saved in the manual testing. Huge profits in automation projects lies in the fact that the problem can be found in proximity to the time the problem was created. It affects the cost of fixing the problem. Methods, like extreme programming use this exact fact.