Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Financial Services | the way we see it

ROI Metrics for Testing Engagement

A Penny saved is a Penny earned

Table of Contents

Prepared By Vineet Singh Pangtey

Date August 2010

1. Introduction

Software testing has often been considered as a necessary evil. It takes great amount of effort to convince people on the importance of testing. The very generic notion about testing is that it does not give any Return on the Investment made and all the investment done is to ensure the quality of the product to be delivered. So, the question is – Is there any ROI (Return on Investment) of Software testing? If yes, how do we measure the ROI in quantitative terms? What should be the approach and different components to be considered for calculating ROI for a testing engagement?

This white paper provides answer to all these questions and explains the approach to calculate the ROI for a testing engagement. It also talks about how to do a comprehensive cost benefit analysis considering all the components of Testing ROI.

Calculating ROI for testing engagement also become important as testing spending forecast is going to be increased as per the NelsonHall report. Exhibits 1 and 2 display forecast for specialist software testing spending and growth broken down per delivery location during the 2007-2013 period as published by NelsonHall on June, 2009.

Exhibit 1: 2007 -2012 Specialist Software Testing Forecast ($m)

	2008	2009	2010	2011	2012	2013
India Delivered testing	2,575	2,700	3,100	3,725	4,475	5,375
Other low-cost location delivered testing	325	350	375	425	500	575
Onshore delivered testing	3,650	3,725	3,950	4,300	4,750	5,225
Total specialist software testing spending	**6,550**	**6,775**	**7,425**	**8,450**	**9,725**	**11,175**

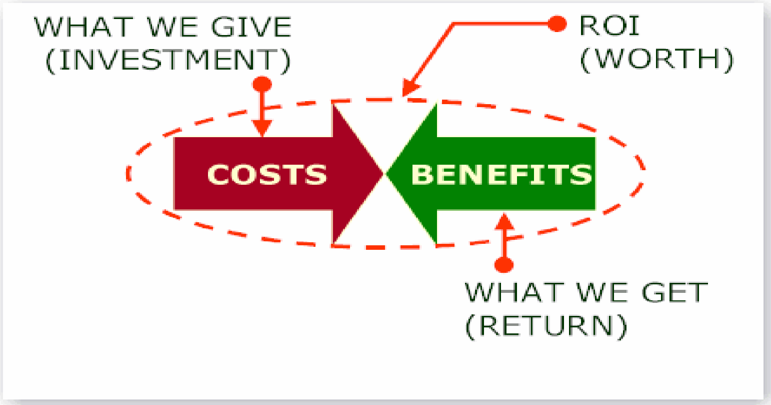
Exhibit 2: 2007 -2012 Specialist Software Testing Growth Forecast (%)

	2008	2009	2010	2011	2012	2013	2008/13 CAGR
India Delivered testing	10	5	15	20	20	20	16
Other low-cost location delivered testing	10	5	10	15	15	15	12
Onshore delivered testing	7	2	5	10	10	10	7
Total specialist software testing spending	**8**	**3**	**9**	**14**	**15**	**15**	**11**

2. ROI – Return on Investment

Return on Investment is the ratio of money gained or lost on an investment relative to the amount of money invested.

Exhibit 3: ROI

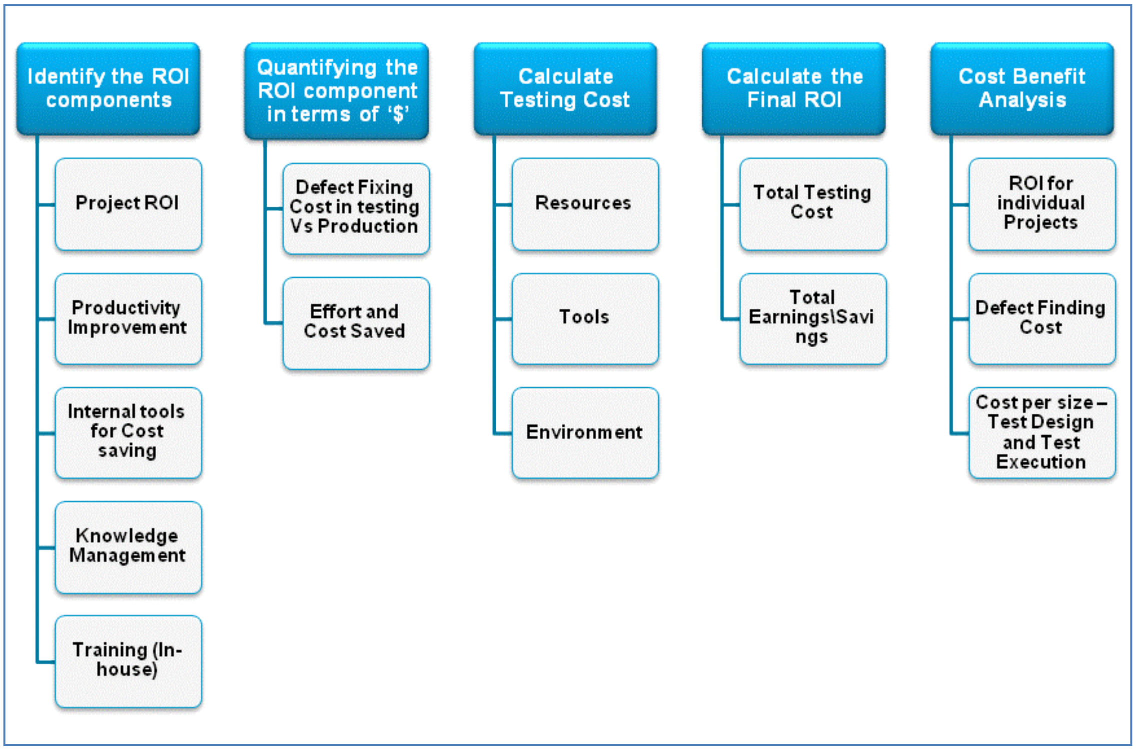


The amount of money gained or lost may be referred to as interest, profit/loss, gain/loss, or net income/loss. The money invested may be referred to as the asset, capital, principal, or the cost basis of the investment

ROI % = ((total earnings - total costs) / total costs) x 100

3. ROI for Testing Engagement – The Approach

The below three steps explains the approach used to calculate the ROI for the overall testing engagement.



3.1. Identify the ROI components

We all know the fact that testing is an investment which does not give any tangible or immediate earnings. So, the overall approach for calculating ROI for a testing engagement is based on a basic thought or principle – A penny saved is a penny earned. The idea is to identify all the components during testing lifecycle which are giving any kind of savings.

Below is the list of all the possible components which can save costs -

- Project ROI
  - Cost saved by raising defects in early stages (Defect Fixing Cost) – This is the main component for calculating the cost saved by any testing team. The whole idea of setting up a testing team is to identify defects in early stages of software development lifecycle. The earliest the defect is identified less is the cost.
  - Automation ROI – This is second biggest component of savings. Automation gives you immediate savings as it reduces the time for test execution. The only thing which is

needs to be considered here is the breakeven point from where the Automation start giving a positive ROI as the higher tool cost pulls the automation as negative in the initial cycles.

- Reusable Components/Assets – All the reused components – test cases, data can be considered under cost saved by reusable components.
  - Test Cases
  - Automation Scripts
  - Test Data
- Productivity Improvement – This component can be considered if there is a service level agreement (SLA) with customer on test case authoring and test case productivity. If the current productivity numbers are more than agreed numbers then it can also be considered for cost savings.
- Internal tools for effort and Cost saving – Any tool, say used for communication, which reduces manual effort can be considered.
- Knowledge Management – All collateral, Proof of Concepts, book of knowledge developed which has not been charged to customer.
- Training (In-house) – All in-house trainings which has not been charged to customer.

3.2. Quantifying the ROI component in terms of '$'

Once all the components have been identified for a testing engagement, the effort/cost saved should be converted in terms of $ to calculate the actual ROI.

3.3. Calculate Testing Cost

The next step in the ROI calculation is to calculate the overall testing cost in terms of effort and resources. The main components would be resources, tools and test environment cost.

3.4. Calculate the final ROI

Once the testing cost is being calculated the final ROI can be calculated using the ROI formula. It would be helpful if we can calculate the testing cost at different levels like for the individual project, testing phases etc for a better cost comparison and drawing conclusions.

3.5. Cost Benefit Analysis

The final step is Cost Benefit Analysis to give the overall approach a holistic view. This can be performed at an engagement level or at project level. The outcome of the Cost Benefit Analysis would be identifying the improvement actions and roadmap to complete the same.

4. Calculating ROI – A Case Study

Once all the components have identified to calculate cost saved in terms of money, the next step is implementing the approach for a testing engagement. In this step, we will calculate the final ROI using the approached mentioned in the section 3 with a case study.

4.1. Components considered

The ROI components can differ from one case to another. In this case, there is no automation implemented and there is no service level agreement for the productivity numbers.

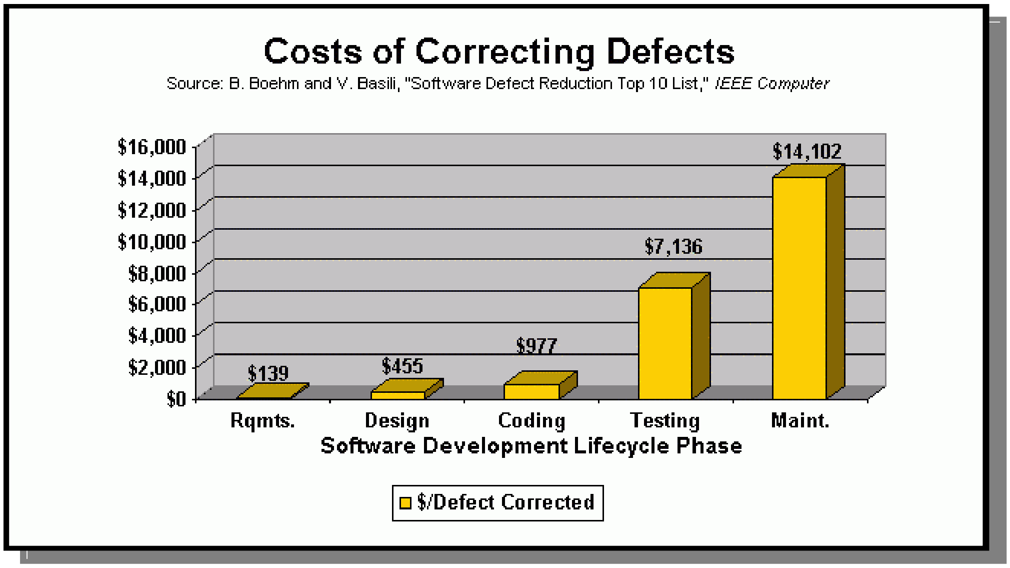
Below is the list of the components used in this particular case:

- Cost saving by raising defects in early stages (Defect Fixing Cost)
- Training - In-house
- Internal tools for time saving
- Reusable Components/Assets
  - Test Cases
- Testing Cost
- Onsite Hrs
- Offshore Hrs (FTA and Tester)

4.2. Defect Fixing Cost

The biggest component of the Testing ROI comes from the savings through identify defects by the testing team prior to production. There are industry standards available on the defect fixing cost for each software development phase by various organizations. We should avoid using these numbers as-is because we are calculating ROI for a particular testing engagement and the particular engagement might not accept or agree with the industry benchmarks.

Exhibit 4: Cost of fixing defect by IEEE



The graphs in Exhibit 4 shows the cost of fixing defect and these numbers are given by IEEE. The defect fixing cost is $ 7,136 in testing and $ 14, 102 in production. This means that the defect fixing cost in production is 1.99 times more than the defect fixing cost in testing.

OR, in other words – Defect Fixing Cost in production/maintenance = 1.99 * Defect Fixing cost in Testing

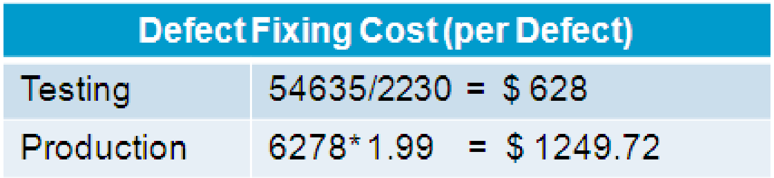
So, we will be using the formula in our ROI calculation. So, the first step we need to do is to identify the defect fixing cost in testing for the particular engagement in testing phase. And then using the ratio we can come up with the defect fixing cost in production using the above formula.

Exhibit 5: Data for Defect fixing Cost

S.N.	Project Name	SIT Defects fixed by Dev Team	No of Working days in SI for Dev team	Team Size	Total No. of Hrs	Billing Cost ($)
1	Project A	68	35	7	1960	48020
2	Project B	11	13	1.25	130	3185
3	Project C	7	11	1	88	2156
4	Project D	1	13	0.5	52	1274
Total		87	72		2230	54635

The table in Exhibit 5 calculates the defect fixing in testing for the particular engagement. It lists all the defects fixed by the development team and their total spent on fixing the defect in terms of $.

Using the above data, the defect fixing cost in testing would be $ 627.99 as given in the table below -



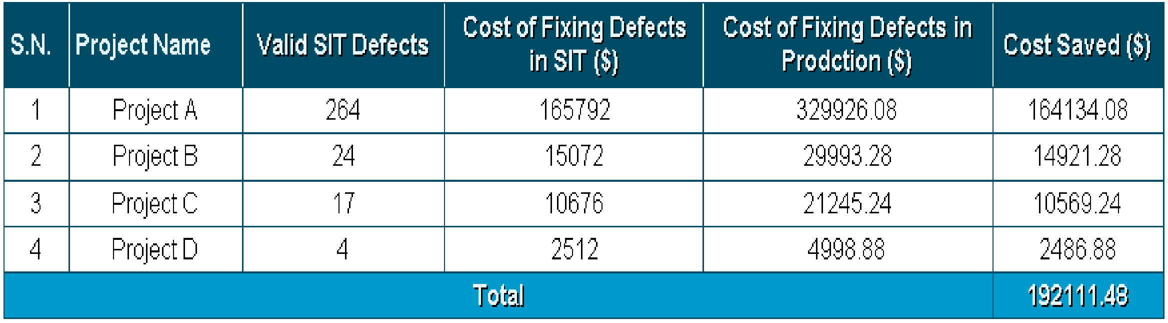
Defect Fixing Cost (per Defect)	
Testing	54635/2230 = $ 628
Production	6278* 1.99 = $ 1249.72

4.3. Quantifying Cost Saving Components

A) Cost saving by raising defect in testing phase

After calculating the defect fixing cost in testing and production for a particular engagement, these numbers can be used to calculate the savings by raising defects during testing phase.

Exhibit 6: Cost saved using defect data



S.N.	Project Name	Valid SIT Defects	Cost of Fixing Defects in SIT ($)	Cost of Fixing Defects in Prodction ($)	Cost Saved ($)
1	Project A	264	165792	329926.08	164134.08
2	Project B	24	15072	29993.28	14921.28
3	Project C	17	10676	21245.24	10569.24
4	Project D	4	2512	4998.88	2486.88
Total					192111.48

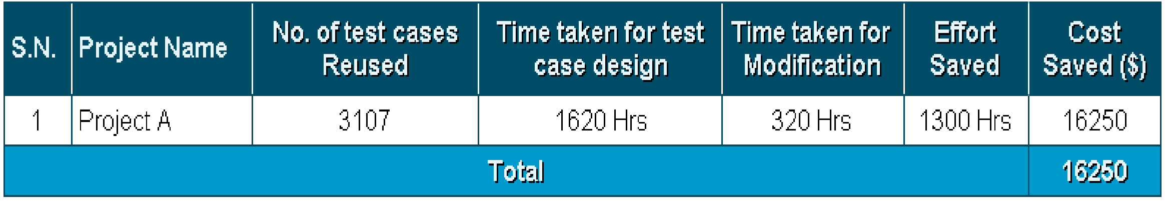
What we are doing here is, calculating the defect fixing cost in testing and production for the same defects found in testing. The idea is that if there is no testing done then the same defect would have to be fixed in production. So, by introducing testing phase we are capturing the defect in an early phase which is cheaper in terms of fixing defects. So, the cost saved by raising defects in testing for a particular project = Cost of Production Defects - Cost of testing defects

Assumption = There is no production defect for all the projects

B) Cost saved by reusable components

In this case, there is only one project – A where the test cases has been reused from previous releases after some small modifications.

Exhibit 7: Cost saved by reusing test cases

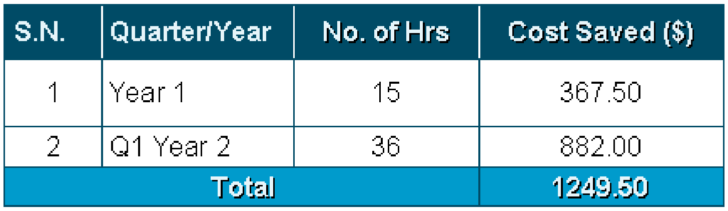


S.N.	Project Name	No. of test cases Reused	Time taken for test case design	Time taken for Modification	Effort Saved	Cost Saved ($)
1	Project A	3107	1620 Hrs	320 Hrs	1300 Hrs	16250
Total						16250

C) Cost saved by training hrs

The table in Exhibit 8 shows the Cost saved by conducting in-house trainings.

Exhibit 8: Training Hours

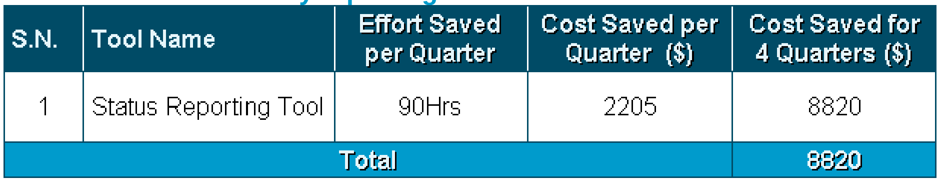


S.N.	Quarter/Year	No. of Hrs	Cost Saved ($)
1	Year 1	15	367.50
2	Q1 Year 2	36	882.00
Total			1249.50

D) Cost saved by tools

The tool used is this case is status reporting which captures the data directly from the Quality Centre and provides the report using macros. The effort saved using the reporting tool per quarter is 90 hrs.

Exhibit 9: Cost Saved by reporting tool

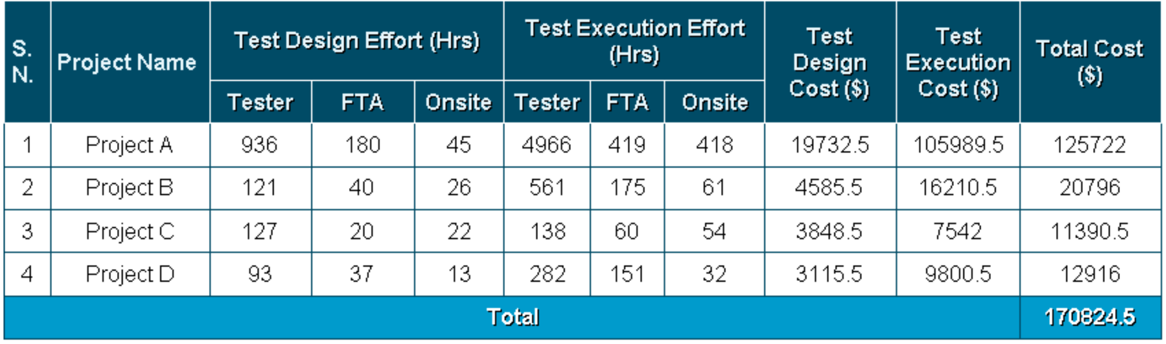


S.N.	Tool Name	Effort Saved per Quarter	Cost Saved per Quarter ($)	Cost Saved for 4 Quarters ($)
1	Status Reporting Tool	90Hrs	2205	8820
Total				8820

4.4. Total Testing Cost

The testing cost has been calculated for each individual project at a testing phase level – Test Design and Test Execution. The cost includes the resource cost at onsite and offshore.

Exhibit 10: Testing Costs

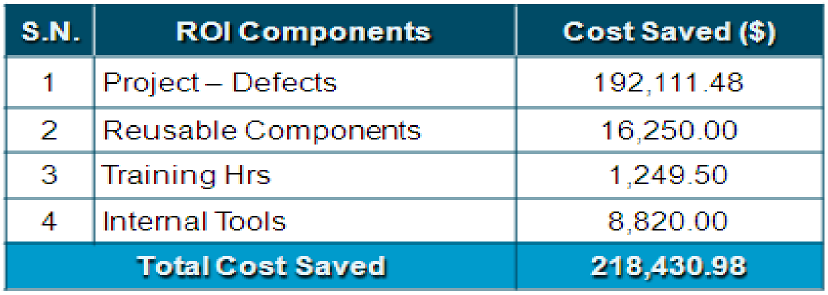


S. N.	Project Name	Test Design Effort (Hrs)			Test Execution Effort (Hrs)			Test Design Cost ($)	Test Execution Cost ($)	Total Cost ($)
		Tester	FTA	Onsite	Tester	FTA	Onsite			
1	Project A	936	180	45	4966	419	418	19732.5	105989.5	125722
2	Project B	121	40	26	561	175	61	4585.5	16210.5	20796
3	Project C	127	20	22	138	60	54	3848.5	7542	11390.5
4	Project D	93	37	13	282	151	32	3115.5	9800.5	12916
Total										170824.5

4.5. Final ROI

The ROI is being calculated at engagement level for all the projects. The table in Exhibit 11 shows all the components considered in calculating ROI and the total cost saved.

Exhibit 11: Total Costs Saved



S.N.	ROI Components	Cost Saved ($)
1	Project – Defects	192,111.48
2	Reusable Components	16,250.00
3	Training Hrs	1,249.50
4	Internal Tools	8,820.00
Total Cost Saved		218,430.98

Total Testing Cost = $ 170,824.5

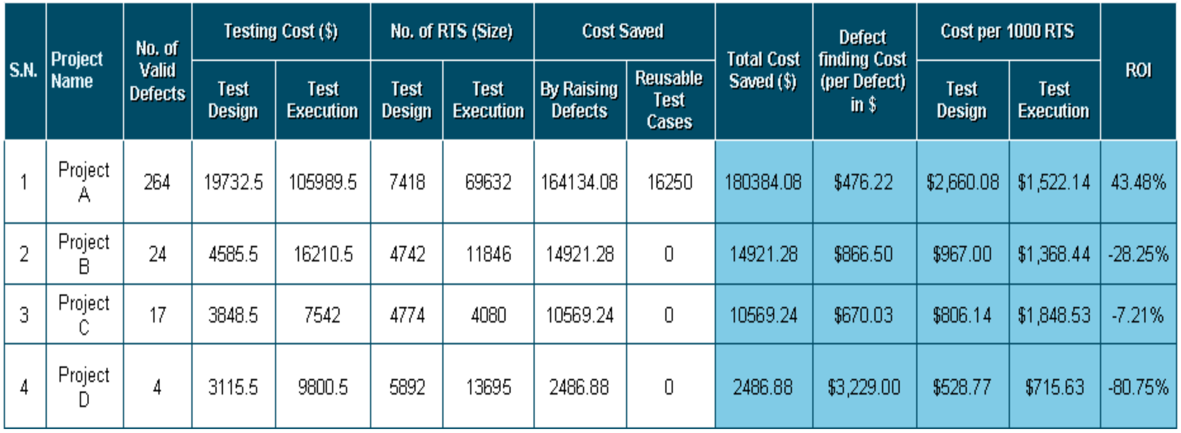
Total Cost Saved = $ 218, 430.98

Final ROI = ((Testing Cost Saved - Testing Cost) / Testing Cost)) * 100

= 27.87 %

4.6. CBA – Cost Benefit Analysis

Exhibit 12: Cost Benefit Analysis



S.N.	Project Name	No. of Valid Defects	Testing Cost ($)		No. of RTS (Size)		Cost Saved		Total Cost Saved ($)	Defect finding Cost (per Defect) in $	Cost per 1000 RTS		ROI
			Test Design	Test Execution	Test Design	Test Execution	By Raising Defects	Reusable Test Cases			Test Design	Test Execution	
1	Project A	264	19732.5	105989.5	7418	69632	164134.08	16250	180384.08	$476.22	$2,660.08	$1,522.14	43.48%
2	Project B	24	4585.5	16210.5	4742	11846	14921.28	0	14921.28	$866.50	$967.00	$1,368.44	-28.25%
3	Project C	17	3848.5	7542	4774	4080	10569.24	0	10569.24	$670.03	$806.14	$1,848.53	-7.21%
4	Project D	4	3115.5	9800.5	5892	13695	2486.88	0	2486.88	$3,229.00	$528.77	$715.63	-80.75%

- ❖ ROI is comparatively less for project D as
  - ➢ It was a complete batch testing for data validation
  - ➢ Code Changes were less but scope of testing was more
- ❖ ROI for these type of projects can be increased by reducing testing cost by below methods
- ❖ Implementing OAT to optimize test coverage
- ❖ Test Design by using tools like Requirement Centre
- ❖ Usage of Data validation tools

5. Conclusion

The above approach gives the ROI for the testing engagement. The final question here is how to use or interpret the data. The point to be noted here is that isolated data or metrics has no meaning. If we say that ROI for year 2 is X percentage, it does not give much insight. The ROI data has to be looked upon in comparison to the previous data. So, it would make sense if we say the ROI for year 2 is X percentage in comparison to year 1 which is Y percentage. The next level analysis would involve identifying the root cause for a lower ROI after the comparison is done and indentify innovations or best practices to reduce the testing cost.

One more important thing is here to look at the ROI data in co-relation with other testing metrics. A positive or high ROI is meaningless if there is too many effort or schedule variance, or if the quality of the product delivered by testing team is not up to the mark.

Below are some techniques/tools for saving testing costs -

1. Test Automation for execution
2. Test Case optimization using Orthogonal Array Technique
3. Test Data Management tools
4. Test Design Automation using Model based Testing
5. Communication and Reporting tools
6. Tools for data comparison

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

www.capgemini.com/financialservices

Capgemini Financial Services
6400 Shafer Court
Rosemont, Illinois 60018
Phone (847) 384 6100, Fax: (847) 384 0500