Outsourcing testing activities – How to prove cost reductions?

H.S. van Heeringen

Abstract

Outsourcing (parts) of the ICT organization is for many organizations a hot item these days. In some cases the entire ICT department is outsourced to a domestic or to a foreign ICT supplier, but most organizations outsource only part of their ICT operations. In many cases this part is either the development part or the enhancement part, but nowadays also a trend can be watched with regard to the outsourcing of all system testing activities. One of the most common drivers in outsourcing is the possibility to realize cost reductions. When outsourcing development activities, these cost reductions are measurable. It is possible to calculate the costs per function point for the development activities in-house and also the price per function point of the outsourcing partner(s) and the difference indicates the amount of cost reductions that can be realized.

When outsourcing test activities, this model will not suffice. Although it is possible to calculate the price per function point for test projects, there is one more important variable. This is the test performance delivered: how many defects have been found compared to the defects that should have been found. While the price per function point for test projects may be lower when outsourcing test activities, it still may be possible that there will be no cost efficiencies. This is due to a larger number of defects than necessary residing in the software and the work that is needed to correct these defects as they occur in acceptance test and in production phase. Moreover, the number of defects that is expected to be found in the testing phase is highly dependent on the way that the development of the project has been carried out. If this was a project with a relatively short duration and with a relatively large team size, the number of defects expected is relatively high. It is therefore not possible to evaluate the testing performance objectively without taking into account the way the project has been carried out.

In this paper a model will be presented that will help organizations to assess the real cost efficiencies that they can gain when outsourcing testing activities. The model benchmarks the real project data against the expected project data based on the results from one of the most widely used project estimation tools: QSM SLIM Estimate [1]. Both the amount of hours spent while testing and the number of defects found will be taken into account. In the paper, a sample project will be followed in order to see how the model works.

1. Introduction

It's getting more and more important to be able to calculate the performance of IT development projects and its costs in an accurate way. In this era of outsourcing and off shoring, it is important for the client organization to know the performance of its IT development to be able to assess whether outsourcing their IT organization (or part of it) is really going to save them money. The supplier side has to know its performance to be able to make a good price quotation to win the contract and also to be able to be profitable.

Nowadays, Sogeti Nederland B.V. [2] (Sogeti) experience shows that more and more large companies are outsourcing parts of their IT development activities. In some cases only development or only testing activities are outsourced, but there are also cases in which both development activities and testing activities are outsourced… and these are often outsourced to two different suppliers. This trend has resulted in an increasing number of test lines in the various Sogeti offices in the Netherlands, where each test line performs the testing activities for a specific client.

Most organizations that are mature enough to use software metrics in their IT development processes are able to state their Project Delivery Rate (PDR), for instance in a certain number of hours per function point needed to realize a piece of software in a specific domain. Usually this PDR is the average or the median value of a number of relevant past projects in their experience database. This PDR can be split into a PDR for the major stages in the project lifecycle, such as for instance design, build, test and implementation. These organizations are usually capable of stating their test performance as a certain PDR in hours per function point and they are also able to benchmark this PDR against for instance the ISBSG database [3]. But is it enough to calculate the PDR of a specific test project and benchmark this against the average or median value of this PDR? Is the number of hours spent on testing a project with a specific functional size really a good indicator of the test performance?

In this paper the model will be presented that is developed and is used by Sogeti to assess the test performance of a project. Because Sogeti uses the QSM SLIM toolsuite in its estimating and performance analysis processes, the model benchmarks the real project data against the expected project data based on the results from one of the most widely used project estimation tools: QSM SLIM Estimate. Before explaining the model, lets first see which are the factors that according to Sogeti are the most important ones in assessing test performance.

2. Factors that influence test performance

When investigating the ways in which the test performance can be assessed, there are a lot of factors that influence test performance delivered by a supplier. In our opinion, the three most important factors are:

2.1. The number of defects found

When performing a test, it is important to find as many defects as possible. However, it is not possible to measure test performance only by the number of defects found. When many defects are found, this may indicate that test performance was very good. However, when a lot of defects are found in the first few months after the system has been released into production, this may indicates that test performance was maybe not that good after all. On the other hand, when only few defects are found in the test, this may indicate that test performance was very poor, but this also may not be true. It could be that there are no more defects and all defects have been found. This means that a good assessment of test performance relies on the performance of the development team and also on the number of defects that reside in the software after the testing phase.

2.2. The number of hours spent

Next to the number of defects found, test performance is also influenced by the number of hours spent on testing activities. How efficient and how productive has the test been carried out? The efficiency is not only the result of the knowledge and skills of the people that are

carrying out the test, but also by the number of defects that are present in the software. Each defect has to be logged, reported, retested and reported again and so leads to extra work.

2.3. The time interval that is available

In most software projects there is a fixed date on which the system has to be implemented in the production environment. A project can roughly be divided into two phases: development phase (including design) and testing phase. It is a well-known fact that development phases of projects tend to last longer than planned. Usually this means that the testing phase has to be carried out in a shorter time period than planned, in order to be able to still manage to deliver the system on the desired date. So, possibly the test team has done all that it could in the reduced testing phase is quite high, but a lot of defects are still found after the testing phase. This means that it is possible to reach a good test performance, even though a lot of defects are found after completing the testing phase.

2.4. Relation between the factors

The question is now: What is the relation between these factors. If the time period available for a testing phase is longer, how many extra defects have to be found? How many extra hours are needed when the testing phase has a shorter time interval. These are complex questions to answer. We are using the QSM SLIM Estimate tool to help us answer these questions.

3. Setting the benchmark norm

Sogeti uses SLIM Estimate to estimate software development projects. After entering a number of parameters, like for instance the functional size, a detailed solution is generated based on experience data in the metrics database. This solution shows among others the number of hours that is most likely to be spent in the different phases, an optimal duration of the project and the maximum team size that is working on the project. But it also predicts the quality of the software delivered. Not only the number of defects expected, but also the expected mean time to defect (MTTD) is generated by the tool. If there is organizational experience data about defects available in the particular organization, the prediction can be made upon this data. In table 3.1 an example of a work breakdown structure is shown with the estimation of the number of hours for systems testing.

Table 3.1: Predictions of system testing hours

Task	Start Date	End Date	Duration (Months)	Hours	Average number of fte
Fase 2: Requirements & Design	1-1-2008	16-6-2008	5,53	2696	2,8
2.1 Functional Design	1-1-2008	16-6-2008	5,53	2292	2,4
2.2 Contract and Project mgm.	1-1-2008	16-6-2008	5,53	404	0,4
Fase 3: Build and test	11-4-2008	6-12-2008	7,86	8551	6,3
3.1 Contract- and Projectmgm	11-4-2008	6-12-2008	7,86	1129	0,8
3.2 Project preparation	11-4-2008	23-4-2008	0,43	103	1,4
3.3 Technical Design	16-4-2008	17-5-2008	1,05	573	3,2
3.4 Development	23-4-2008	24-10-2008	6,04	5045	4,8
3.5 Systems test	**1-9-2008**	**24-10-2008**	**1,77**	**1248**	**4,1**
3.6 Support Acceptance test	24-10-2008	6-12-2008	1,45	453	1,8

The quality of the software delivered relies heavily on the way the project is carried out. Especially the duration and the maximum team size are important factors that influence the

quality of software. When there is a lot of time pressure (short duration), the quality of the code delivered is the lowest. An example of a prediction is shown in the next table.

Table 3.2: Predictions of software quality

Duration Development and test (months)	Max. development and test team size (# FTE)	Hours needed for Test phase	Defects in systems test	Defects in acceptance test	Defects first month of production
13,6	6	4.154	68	37	8
12,8	8	5.232	75	42	12
12,2	10	6.258	85	50	18
11,8	12	7.244	98	60	26

When the size of the software and the productivity of the team are constant, duration is the most important factor that influences software quality. A relatively short duration means that the project has to be carried out by a relatively large team. The effect that then takes place can be seen in the table above. Software quality is deteriorating, even though more testing hours are spent. Most organizations are not aware of this fact. Usually they understand that if a system has to be build in a short time period, this means it will get more expensive. They usually don't recognize the fact that software quality will be lower and that will lead to more maintenance costs after the project has been released into production phase.

SLIM Estimate provides the possibility to simulate the project after completion, in a way that it's possible to estimate the number of defects and the number of testing hours based on the actual hours spent and the actual duration of the project. The estimate shows for instance the amount of hours that should have been spent in the testing phase, the number of defects that should have been found in systems testing and the number of defects that should have been found in acceptance testing. The number of defects found and the time frame in which they are found are translated by the tool into a specific *Defect Tuning Factor*. This estimation, based on the actual project data, is in the proposed model the baseline against which the test performance can be benchmarked.

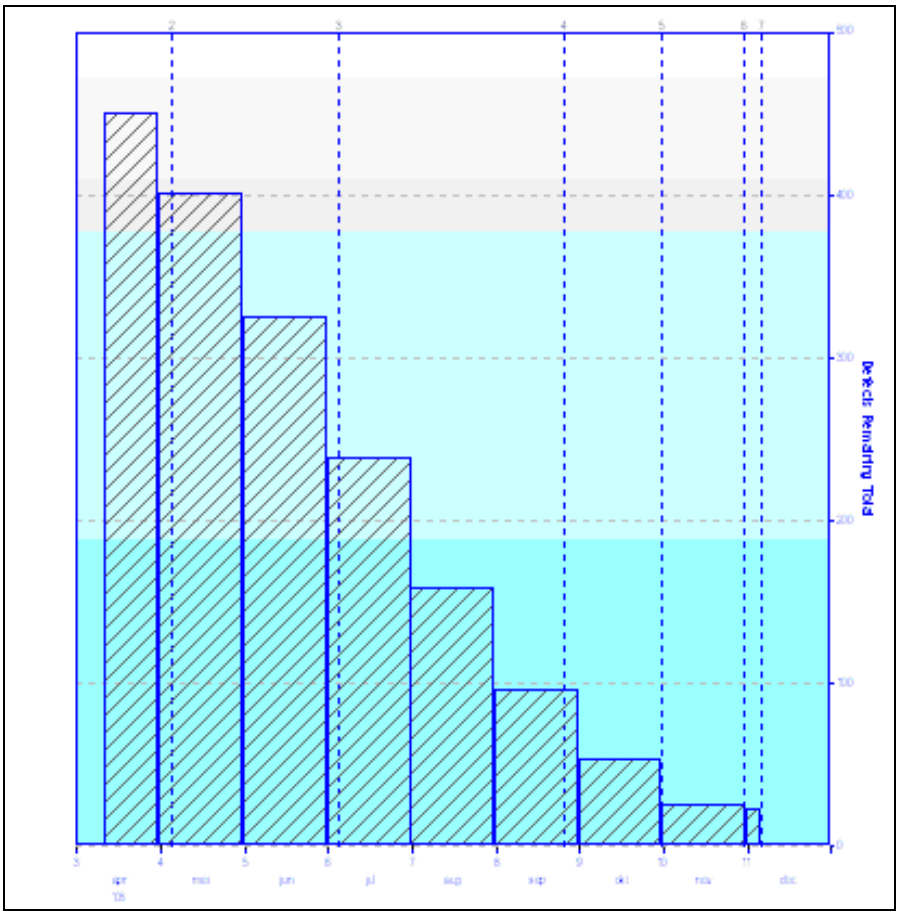

Figure 3.1: Defects that are residing in the software per month and per milestone (vertical dotted lines). Milestone 4 is the end of the development phase, milestone 5 is the end of the systems test and milestone 6 is the end of the acceptance test.

4. Test performance

After setting the benchmark, based on your own experience data and the project actual data of the project completed, the following data is present:

- the number of hours that was expected
- the number of hours actually spent
- the number of defects found expected
- the number of defects actually found
- the number of defects expected to be found after completing systems testing
- the number of defects actually found after completing systems testing

With this information it is possible to assess in a number of cases whether systems testing was better or worse than expected. Table 4.1 shows the different scenario's.

Table 4.1: Test performance assessment

	Less defects found than expected	Defects found equal to expectation	More defects found than expected
Less hours spent than expected	**?**	**+**	**++**
Hours spent equal to expectation	**-**	**±**	**+**
More hours spent than expected	**--**	**-**	**?**

Whether the question marks become a plus or a minus depends on the view of the client. When cost savings are more important than defects found, the question mark in the upper left corner could become a plus as well. When finding defects is more important than hours spent, the question mark in the lower right corner will become a plus.

So, it is possible to assess the systems test performed. But how can we use this insight to calculate the cost reductions that we have delivered in a contract. First we have to find out what the test performance is of the client before the outsourcing deal. This is step 1 in the model.

5. Model step 1: Setting the baseline of the client

In this paper we are looking at a fictional contract that Sogeti and a major client (client X) have agreed upon. The contract states in short that the client outsources all systems testing activities to Sogeti and Sogeti committed to realize cost reductions of 10 percent in the first year and another 10 percent in the second year. The client had outsourced the development activities already to another ICT supplier and the client is in charge of acceptance testing and implementation activities themselves

Sogeti uses the following model when insourcing testing activities of a specific client.

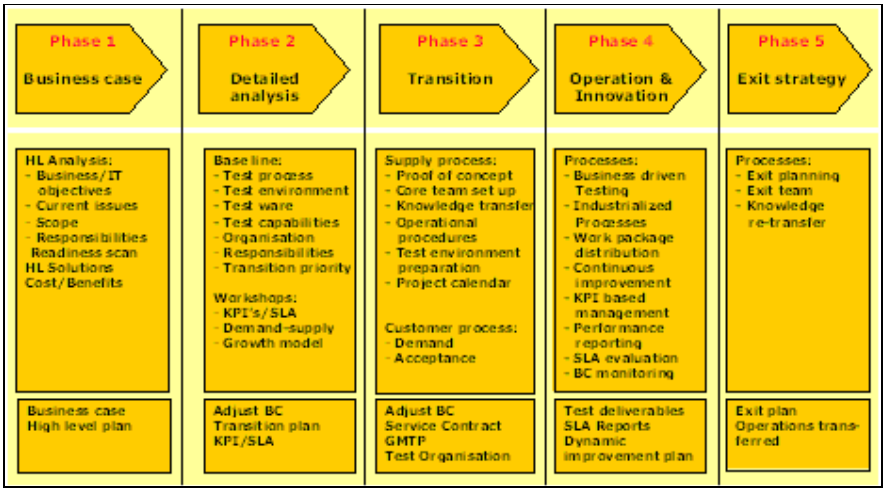

Figure 5.1: Sogeti Outsouring model for test services

To be able to make cost reductions visible, first the performance of the client before outsourcing has to be assessed in an objective way. To do this, a baseline analysis has been carried out for client X. This has taken part in Phase 2 of the model described.

For client X the following approach has been used:

1. A sample of 5 recent projects has been identified to include in the baseline
2. These projects were sized using the COSMIC method [4]
3. The actual performance of the projects has been administrated in SLIM Estimate
4. The results are written in a baseline report
5. Client X had to agree on this baseline report and, by doing this, committed to calculate cost reductions based on this baseline

The baseline report shows the analysis of the different projects. The most important results of this baseline are:

- Detailed analysis per project
- SLIM Estimate template Work Breakdown Structure
- Average PI (productivity index) of the clients projects
- Defect tuning factor of the clients projects
- Costs per testing hour

Because of the fact that we have to prove cost reductions, it is important to define costs per testing hour in an objective way. With the client was agreed to use the clients internal administrative hour rate for testing activities in order to be able to put a price tag on a testing hour. The choice on which method to use for measuring the size of the software should be made together with the client. Possibilities are the functional size measurement methods COSMIC, IFPUG [5] or NESMA [6] FPA or effective source lines of code. The latter is the most easy to collect and is also the best usable in SLIM Estimate.

For client X, the baseline report showed among others the following data

Table 5.1: Baseline data Client X

Project	Size in CFP	Hours spent in Test phase	Defects found in systems test	Defects in acceptance test + 1st month Production
A	310	554	12	8
B	512	937	27	5
C	212	508	22	12
D	224	732	52	18
E	487	878	35	8

The cost per test hour was set to 80 euro.

6. Model step 2: Operational execution

The baseline report shows the exact costs and performance of the clients testing situation before outsourcing. After the transition phase, the performance of Sogeti has to be measured. Because of efficiency reasons, the client and Sogeti agreed not to measure every single project, but in advance a number of projects were selected to measure and it was agreed that these projects would be used to demonstrate the cost reductions for the client.

For every project that is to be measured, the following four steps have to be carried out:

6.1. Step 1

The size of the product to be developed has to be measured using the agreed method. In the case of client X the COSMIC method was used.

6.2. Step 2

The project is estimated with SLIM Estimate and a calculation is made using the parameters of the client in the baseline together with the actual duration of the project. The number of testing hours that the client would have spent is expected, together with the number of defects that the client would have found in the duration actually realized. The clients Work Breakdown Structure and the clients' defect tuning factor are used to be able to make a realistic prediction of the project in the case the client had not outsourced its testing activities. The results are shown in table 6.1

Table 6.1: Project Q estimate based on baseline data

Scenario	Test hours TU	Defects S-test FS	Defects A-test + production FAP	Defects total FT (=FS+FAP)	Ratio FS/FT	Hour rate R	Costs C (TU*R)
Client baseline	2.045	30	31	61	0,49	80	163.600

This can only be done 1 month after the system has been released into production, as the number of defects in the first month is a variable here.

6.3. Step 3

The actual project data is put in the table, which results in table 5.2

.

Table 6.2: Project Q baseline and actual data

Scenario	Test hours TU	Defects S-test FS	Defects A-test + production FAP	Defects total FT (=FS+FAP)	Ratio FS/FT	Hour rate R	Costs C (TU*R)
Client baseline	2.045	30	31	61	0,49	80	163.600
Sogeti actuals	1.800	35	30	65	0,53	82	147.600

In this project, less hours are spent compared to the baseline and relatively more defects are found compared to the defects found after the systems test. Test performance delivered by Sogeti in this project can therefore be assessed as well.

6.4. Step 4

After step 3, we have to assess whether the test performance of the measured projects is really resulting into the cost reductions promised in the contract. To do so, two important metrics are calculated: *Outsourcing effectiveness* and *Outsourcing efficiency*.

Outsourcing effectiveness (OV) is a metric that indicates whether relatively more defects are found than the client would have, or not. The ratio between the defects found in systems test (FS) and the total number of defects found after the code was released for the systems test (FT) is calculated for both the baseline scenario as the actual scenario. The Outsourcing effectiveness is the relative difference between the ratio FS/FT in the baseline scenario and the one in the actual scenario.

Outsourcing effectiveness (OV):

$$\frac{\text{Actual (FS / FT) - Baseline (FS / FT)}}{\text{Baseline (FS / FT)}} * 100\%$$

In project Q, Sogeti has realized an OV of (0.53-0.49)/0.49 ≈ **8,2 %**

Outsourcing efficiency (OF) is calculated by the relative difference in the number of testing hours times the hour rate.

Outsourcing efficiency (OF):

$$\frac{\text{Baseline C – Actual C}}{\text{Baseline C}} * 100\%$$

In project Q, Sogeti has realized an OF of (163.600–147.600) / 147.600 ≈ **9,7%**

In this model, these two metrics together are the basis for the metric Cost Reduction (CR) for the systems testing phase.

Cost reduction has a relation with Outsourcing effectiveness and with outsourcing efficiency. This relation is made visible in figure 6.1

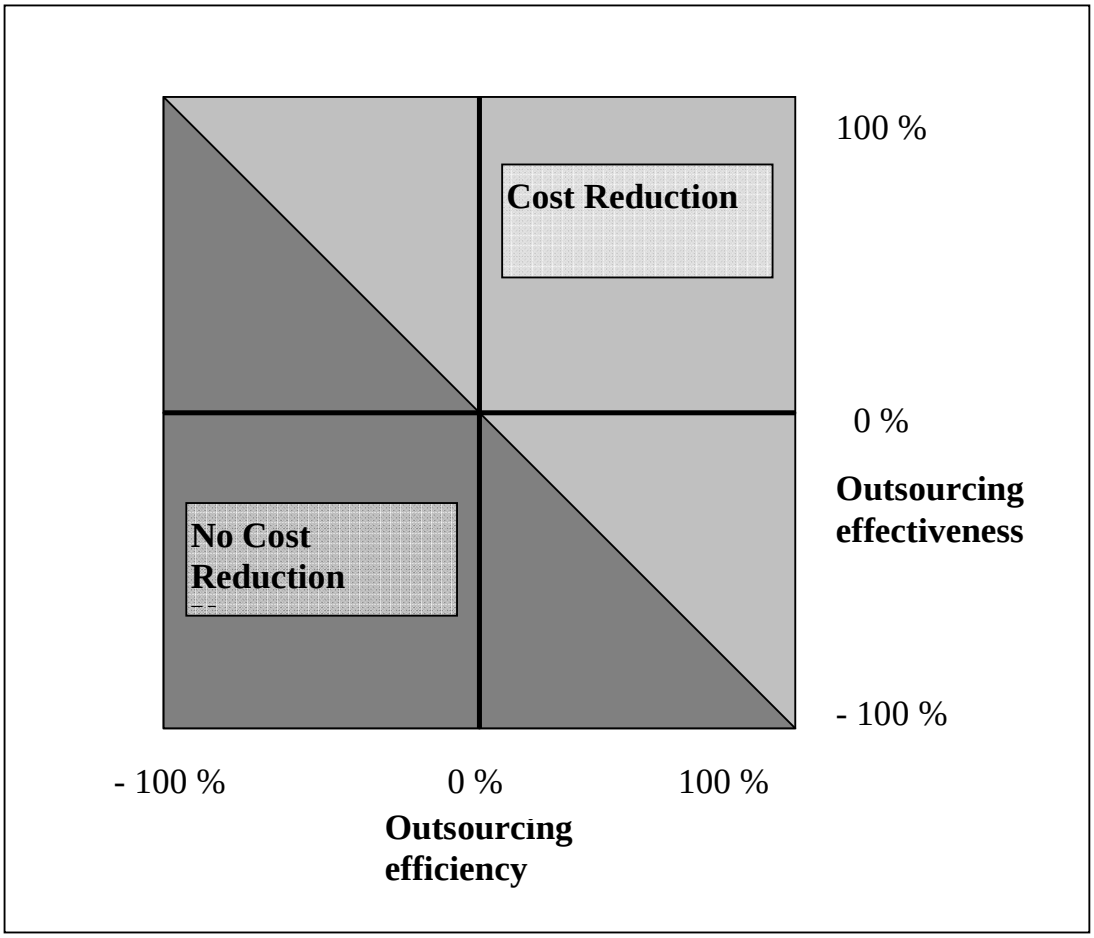

Figure 6.1: Cost Reduction expressed in terms of Outsourcing Effectiveness and Outsourcing Efficiency

In the top right of figure 6.1 the cost reduction is clear. Relatively more defects have been found compared to the baseline and relatively less hours have been spent. In the bottom left it's clear that the performance was quite bad.

In this model, the assumption has been made that the relation between outsourcing efficiency and outsourcing effectiveness is 1 to 1. This means that for a client both metrics are equally important. If this is not the case, this has to be agreed upon in the contract negotiation phase between the two parties. If outsourcing efficiency is more important, the diagonal should shift by putting weight factors to the metrics.

This leads to the following formula for Cost Reduction:

Cost Reduction

CR = (w1 * OV) + (w2 * OF) / w1+w2

CR = Cost Reduction
OV = Outsourcing Effectiveness
OF = Outsourcing Efficiency
w1 = weight factor OV
w2 = weight factor OF

in table 5.3, a number of scenario's have been displayed in order to make this visible. A negative CR means that the costs of outsourcing are not reduced at all, but are even higher than before outsourcing.

Table 6.3: A number of scenario's and the calculated Cost reduction

Scenario	OV	w1	OF	w2	Cost Reduction
1	25 %	1	- 30%	1	(25% + -30%) /2 = - 2,5%
2	25 %	2	30%	1	((2*25%) + 30%) /3 = 26,67 %
3	- 25 %	1	- 30%	2	(-25 % + (2*- 30%)) /3 = -28,33 %
4	- 25 %	4	30%	7	((4*-25%) + (7* 30%)) / 11 = 10 %

When applying this model, it is possible to assess cost reductions per project. After all the projects that were identified as the projects to be measured actually have been measured this way, step three of the model can be performed.

7. Model step 3: Evaluation

After the contract period has ended we can see if the percentage of cost reductions, that were agreed upon in the contract, have been realized or not. This can be done by filling in table 7.1 with the data gathered. For client X the following projects were measured.

Table 7.1: Cost reduction in the contract

Project name	Outsourcing effectiveness	Outsourcing efficiency	Test Costs baseline	CR	CR in EUR
Q	9,7%	8,2%	€ 163.600	8,95%	€ 14.642
R	12,3%	6,1%	€ 312.150	9,2%	€ 28.718
S	8,2%	3,0%	€ 122.650	5,6%	€ 6.868
T	6,4%	13,3%	€ 718.120	9,85%	€ 70.735
U	15,6%	8,8%	€ 67.050	12,2%	€ 8.180
TOTAL			**€ 1.383.570**		**€ 129.143**

The overall cost reductions in this contract were 129.143 / 1.383.570 ≈ 9,3%. While substantial cost reductions have been demonstrated, it was not enough to fulfill the promises in the contract, which was 10 percent in the first year.

8. Conclusions & Discussion

Many organizations are nowadays involved in outsourcing parts of their ICT department. When outsourcing testing activities, these organizations usually want the ICT supplier to agree on specific percentages of cost reductions. To agree on these percentages is usually not a problem, but proving afterwards that the goals are actually reached usually ís a big problem. In this paper the model is described that Sogeti uses to prove cost reductions for clients that outsource their testing activities to Sogeti. Before the outsourcing is effectuated, a baseline analysis is made in which the test performance of the client is established. During the contract period a number (or all) of the projects are measured and the test performance of the supplier is measured. The formula's for outsourcing efficiency, outsourcing effectiveness and cost reductions can then be calculated and in the end the percentage and amount of cost reductions for the client can be calculated. This model helps both the client and the supplier to assess cost reductions in outsourcing test activities and the first experiences with the model are very positive.

However, new experiences with the model may lead to desires to deepen it. Examples could be:

- Putting thresholds to Outsourcing efficiency and/or effectiveness. E.g. client may not accept an outsourcing effectiveness lower than 5%.
- Specify defects to category and put weight factors to the categories.

References

[1] QSM Software Software Lifecycle Management toolsuite, http://www.qsm.com.

[2] Sogeti Nederland B.V., http://www.sogeti.nl; http://metrieken.sogeti.nl.

[3] ISBSG database, version 10, www.isbsg.org.

[4] ISO/IEC19761:2003, Software Engineering -- COSMIC -- A Functional Size Measurement Method, International Organization for Standardization - ISO, Geneva, 2003.

[5] IFPUG, "Function Point Counting Practices Manual, version 4.2, International Function Point Users Group, 2004, http://www.ifpug.org.

[6] NESMA, "Definitions and counting guidelines for the application of function point analysis A practical manual, version 2.2", Netherlands Software Measurement user Association, 2004 (in Dutch), http://www.nesma.org.