1. INTRODUCTION

Application performance is the ability to perform some or all of the application functions for specific, measurable, pre-defined amounts of time. These amounts could be defined as part of business requirements and could vary widely by application function. For example, an organization can define an expectation of 10 seconds for refreshing a control panel and 20 minutes for executing a monthly report.

When an operation is completed within the specified time we say that the application performs well, when it is over, it performs poorly.

Performance improvement is a process of identifying the root causes of performance loss and implementing sets of solutions toward eliminating them. Ideally, performance improvement projects are triggered by the problem management for specific service.

I am positive that there are studies out there, prepared by reputable organizations, about the impact on productivity due to a poor application performance. If you want to get the scientific explanation, please, go ahead and read them. Otherwise, just look around what you and your co-workers do when you wait for your application to get you to the next step.

Performance problems are bad. People could do much better and achieve much higher results if they are not constantly irritated waiting for that next step screen so they can continue doing what they do best - their jobs.

2. THE LAWS

Nicky's First Law for Application Performance: Once the application performance goes down, it will never recover by itself.

Nicky's Second Law for Application Performance: In static environment the application performance can only change to worse.

3. JUSTIFICATION

Like in every other aspect of doing business here we work with gain and cost. Balancing these two indicators can significantly improve the ROI from particular service.

Let's analyze, for example, what is the cost of specific time for refreshing of a control panel. We can use the monetary value for the "time wasted" to establish a value that we can later compare to the resources needed to bust the application performance to the desired levels. Here is the scenario:

Our organization has IT support personnel of 20 employees. As average each of them is assigned and resolves 10 incident reports every day. All employees monitor their control panels for assigned work, they refresh it approximately 25 times per day. We also know that each employee costs our organization $100/hr.

Let's say that our business requirement is to have the control panel refreshed in 10 seconds. We assume that during this waiting time our employees are doing nothing but getting annoyed waiting, so, from this perspective this is "wasted time".

> 20 employees x 25 times x 10 seconds x 240 business days = 1,200,000 seconds (333.33 hrs)

Our goal is that the cost of "wasted time" for this operation is $33,333 per year. We accepted this "wasted time" as given.

Now let's review the current findings: we found that the actual time for refreshing this control panel is 30 seconds in average.

Following the example above we can calculate the actual cost of "wasted time" for this operation, it is $100,000 per year.

The difference between these costs (actual - goal) is $66,666. We will call it additional cost due to poor performance.

Analyzing this problem we discovered that the performance of this function can be significantly improved if we re-design the query extracting the results. An estimate is made that it will take 60 hours to implement a change that will replace the current query with a new one. Let's accept for the purposes of our example that developing. testing and deployment work cost $150/hr to our organization.

We can now calculate the cost of this change.

> 60 hrs x $150 = $9,000

Now let's compare the additional cost due to poor performance to the amount needed to resolve the problem

$9,000 to resolve < $66.666 to save

I agree that this example is naive and things are much more complex and complicated in real life. The point is that there is a strictly significant way to justify certain changes or reject others.

4. CONCLUSION

The performance problems, once present, can only get worst. They have to be addressed systematically and resolved completely. You should plan for fine tuning of your applications from the very beginning and adopt the ITIL's service model. Do you remember the service lifecycle? Continuous service improvement is an important step toward delivering better services and performance improvement is just a brilliant example of it.