Information Balance

***A Practical Framework for*:**

Measuring ROI of an APM investment

Gabor Szirmak
President
Information Balance, Inc.

White Paper
March, 2009
© All Rights Reserved

A Practical Approach to Measuring ROI on APM Investment

The promise of Application Portfolio Management (APM) is to provide comprehensive insight into an organization's application portfolio. This insight is captured in the institutionalized knowledge base that the APM tool implementation delivers. That knowledge base, coupled with the functionality that is the staple of such tools (inventory, search, visualization, impact analysis, complexity assessment, etc.), offer help with a spectrum of day-to-day development and support tasks. APM allows these tasks to be performed faster and with more accuracy than before.

APM is a Productivity Solution

In a relatively well organized and mature IT shop, APM by itself offers very little, if any, *new* capabilities. But when incorporated into existing work patterns and practices, it contributes to faster turnaround and increased accuracy and precision for many tasks faced by the average IT worker.

Given that APM is fundamentally a productivity solution, most CIO-s will insist on a strong business case before committing to any spending. As net new funding is rarely available for such acquisitions, a complete return of the investment within one or two budget cycles must be demonstrated.

APM vendors and industry analysts often suggest productivity savings in the 10 to 35 per cent range. When APM is fully implemented and rolled out to IT staff, management should see significant savings in their application support budget. How significant? Although difficult to predict accurate savings, it could be 10 per cent, 15 per cent or even 30 per cent. How does one measure the extent of actual savings post-implementation? Or quantify the return on investment?

When investing in a productivity solution, benefits can be both tangible and intangible. A good ROI calculation focuses on the tangible benefits, and only on those that can be measured.

© March 2009

Measuring Productivity Savings – A Simple Approach

A simple approach to measuring productivity savings involves comparing maintenance spending for a given (set of) application(s) before and after the deployment of an APM solution. At best, this approach provides only a cursory measure. How much of the realized savings can be attributed to the introduction of APM? For example, if there was a 20 per cent reduction in maintenance spending, was it because of increased productivity or because of a shift in the pattern and nature of expanded maintenance work? Perhaps other tools were introduced, or there was a change in staff deployment, resulting in increased skills and experience at reduced cost. How does one attribute any portion of that savings to a single factor when compared to the many factors that influence overall productivity?

Measuring Productivity Savings – Task Cycle Time

A more accurate approach to measure productivity savings is to use task cycle time as the basis for assessing changes in productivity.

Most of the typical tasks in an application maintenance environment are routine and repetitive. For example, a change request that necessitates the expansion of a database table requires the same type of impact assessment every time it happens: what programs, copybooks, views, JCL, proc, are affected? If an analyst has performed a task numerous times, she will have a well defined process that repeats itself for similar tasks. She will also know, fairly accurately, how long that task takes. That is what we refer to as task cycle time.

A Practical ROI Framework

The following Activity Tracking Form - Figure 1 - offers a framework for assessing productivity savings that arise from the implementation of an APM solution such as ARM[1]. It is particularly effective in a large, complex legacy environment. This assessment is based on comparing task cycle times before and after the implementation of ARM.

[1] Application Road Map (ARM) is Information Balance's APM solution. For more information visit www.infobal.com.

© March 2009

The first step is to categorize typical tasks that IT staff perform. This provides for the same treatment of similar tasks, and the clean comparison of different tasks. The Activity Codes in Figure 1 represent these task categories.

If we analyze productivity gains by job or function, staff roles need to be categorized as well. Possible categories are seen in the top left corner of Figure 1.

Name:

Role (please select one)

- ☐ Architect
- ☐ Business Analyst
- ☐ Business User
- ☐ Programmer Analyst
- ☐ Tester
- ☐ >>>OTHER

Activity Codes

(1) Analyze/Understand Application Structure
(2) Analyze/Understand Batch Cycle
(3) Analyze/Understand Call Structure
(4) Analyze/Understand Component Tree
(5) Analyze/Understand Program Structure
(6) Determine Test Coverage
(7) Gather / Analyze Application Metrics
(8) Impact Assessment - Code Change
(9) Impact Assessment - Database Change
(10) Impact Assessment - Screen Change
(11) Impact Assessment - Where Used / Referenced
(12) Trace Data Flow (File Usage)
(13) Trace Data Transformation
(14) Trace Relationship Path
(15) Where Used?
(16) >>>OTHER

	Date	Activity (Please use Code)	Task Description (Describe task in some detail)	Effort To Complete Task (HRS)		Number of times you would typically perform this type of task in a month
				Using ARM	Traditional Approach	
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						

Figure 1: Activity Tracking Form

Once the task and role categories are established, several Activity Tracking Sheets - see Figure 2 - need to be completed. Select a representative group of IT staff and ask each person to track their day-to-day activities for a period of perhaps a week. If 20 to 25 people participate and contribute 15 to 20 entries each, that will provide enough data. With very little intrusion, one can collect 3 to 500 data points within a week. Figure 2 illustrates a typical filled-in sheet.

© March 2009

Name: xxxxxxxxxxxx

Role (please select one)

x	Architect
	Business Analyst
	Business User
	Programmer Analyst
	Tester
	>>>OTHER

Area (please select one)

	Production Support
	Enhancements
	Product & Pricing
x	>>>OTHER

Activity Codes

(1)	Analyze/Understand Application Structure
(2)	Analyze/Understand Batch Cycle
(3)	Analyze/Understand Call Structure
(4)	Analyze/Understand component relationships
(5)	Analyze/Understand BMT
(6)	Analyze/Understand Program Structure
(7)	Determine Test Plan
(8)	Gather / Analyze Application Metrics

	Date	Activity (Please use Code)	Task Description (Describe task in some detail)	Effort To Complete Task (HRS)		Number of times you would typically perform this type of task in a month
				Using ARM	Traditional Approach	
1	3-Apr	3	Confirm/document subroutines called by PTS012	0.25	1.50	3.00
2	3-Apr	5	Confirm/document transactional vs. BMT tables used by PTS012	1.00	2.00	1.00
3	5-Apr	10	CRUD Analysis - COVT01, COVERS, COVRRI	0.25	0.50	4.00
4	6-Apr	2	Batch PHUD1210, PHUD1250 comparative analysis of JCL	1.00	2.00	1.00
5	10-Apr	3	C1S720 subroutine call analysis	0.25	1.00	3.00
6	10-Apr	3	C1O886 subroutine call analysis	0.25	1.00	3.00
7	12-Apr	10	CRUD analysis - C1O886 DB2 table usage	0.50	2.00	4.00
8	15-Apr	6	TIO002 - program structure	0.50	3.00	8.00
9	15-Apr	10	Index Lookup/Analysis - BPT032 & DPT032	0.25	0.50	8.00
10	18-Apr	5	Programs using COVCTL	0.10	1.00	1.00
23	23-Apr	2	Annual/Quarterly Batch jobs, how are they set up?	0.25	3.00	1.00
24	26-Apr	9	TRS013 code change impact analysis	0.50	1.00	4.00
25						
26						
27						
28						
29						
30						
31						
32						
33						
34						

Figure 2: Filled-in Activity Tracking Sheet

From the raw data, productivity improvement is easily calculated by activity type, by role and overall based on average savings in task cycle time. Depending on the scope of the survey, findings in the 35 to 85 per cent range are common. For example, if the scope of the survey mostly included analysts engaged in feasibility assessment, impact analysis, or similar activities that demand extensive application component search, relationship tracing, and similar activities, the improvement will be at the high end.

The next step is to quantify the results. Figure 3 illustrates a segment of a result sheet augmented with expense data.

© March 2009

Activity Type	Performed By	Effort To Complete Task (HRS)		Number of Times Task Performed in a Typical Month by a Single Resource	Productivity Gain	Effort Saved (Man-Days) (Single Execution)	Cost Saved ($$$) (Single Execution)	Effort Saved (Man-Days) (Annual)	Cost Saved ($$$) (Annual)
		Using ARM	Traditional Approach						
Analyze/Understand Call Structure	Architect	0.25	1.50	3	83%	0.17	$100.67	6.0	$3,624.00
Impact Assessment - BMT	Architect	1.00	2.00	1	50%	0.13	$80.53	1.6	$966.40
Impact Assessment - Database Change	Architect	0.30	0.50	4	40%	0.03	$16.11	1.3	$773.12
Analyze/Understand Batch Cycle	Architect	5.00	10.00	2	50%	0.67	$402.67	16.0	$9,664.00
Analyze/Understand Call Structure	Architect	0.25	1.00	3	75%	0.10	$60.40	3.6	$2,174.40
Analyze/Understand Call Structure	Architect	0.25	1.00	3	75%	0.10	$60.40	3.6	$2,174.40
Impact Assessment - Database Change	Architect	0.50	2.00	4	75%	0.20	$120.80	9.6	$5,798.40
Analyze/Understand Program Structure	Architect	0.50	3.00	8	83%	0.33	$201.33	32.0	$19,328.00
Analyze/Understand Application Structure	Architect	0.25	1.00	20	75%	0.10	$60.40	24.0	$14,496.00
Analyze/Understand Program Structure	Architect	0.50	3.50	20	86%	0.40	$241.60	96.0	$57,984.00
Impact Assessment - Database Change	Architect	0.25	1.00	2	75%	0.10	$60.40	2.4	$1,449.60
Analyze/Understand Application Structure	Architect	0.20	1.00	50	80%	0.11	$64.43	64.0	$38,656.00
Impact Assessment - Code Change	Architect	1.50	2.00	10	25%	0.07	$40.27	8.0	$4,832.00
Impact Assessment - Where Used / Referenced	Programmer Analyst	0.01	0.16	5	94%	0.02	$12.08	1.2	$724.80
Analyze/Understand Batch Cycle	Programmer Analyst	0.03	0.20	2	85%	0.02	$13.69	0.5	$328.58
Analyze/Understand BMT	Programmer Analyst	0.01	0.20	5	95%	0.03	$15.30	1.5	$918.08
Analyze/Understand Batch Cycle	Programmer Analyst	0.01	0.20	2	95%	0.03	$15.30	0.6	$367.23
Analyze/Understand BMT	Programmer Analyst	0.10	0.30	2	67%	0.03	$16.11	0.6	$386.56
Trace Data Flow (File Usage)	Programmer Analyst	0.10	0.30	3	67%	0.03	$16.11	1.0	$579.84
Impact Assessment - BMT	Programmer Analyst	0.10	0.50	4	80%	0.05	$32.21	2.6	$1,546.24
Determine Test Plan	Programmer Analyst	0.30	1.00	2	70%	0.09	$56.37	2.2	$1,352.96
Trace Data Flow (File Usage)	Programmer Analyst	0.10	0.30	3	67%	0.03	$16.11	1.0	$579.84
Analyze/Understand component relationships	Business Analyst	1.00	3.00	1	67%	0.27	$161.07	3.2	$1,932.80
Impact Assessment - Code Change	Business Analyst	0.50	1.50	2	67%	0.13	$80.53	3.2	$1,932.80
Impact Assessment - Where Used / Referenced	Business Analyst	0.50	2.00	4	75%	0.20	$120.80	9.6	$5,798.40
Impact Assessment - BMT	Business Analyst	0.50	1.00	2	50%	0.07	$40.27	1.6	$966.40
					66.57%				**$197,164.93**

Figure 3: Quantified Productivity Savings

The total productivity gain (66.57 per cent) represents the overall average productivity gain *for the survey group* and *for the tasks that were tracked*. The corresponding monetary savings ($197,165) is also for the same scope. This, of course, is an incomplete picture. Each of the participants likely performs a variety of other tasks in addition to those tracked here, and secondly, there are other staff roles within a typical IT department (programmers, DBA-s, QA people, project management personnel, administrative support, etc.) not represented in the survey. These limited-scope survey group results must be extrapolated to the complete development lifecycle, including all development tasks and full staff complement.

To normalize these results, two steps need to be taken. For each participating group, one must determine precisely what scope the survey covered from the SDLC (task distribution), and to what extent the typical tasks were tracked (task coverage). Then the normalized productivity gain is calculated for each phase and each staff group.

© March 2009

In Figure 4 below, the Architect role experienced a 71.73 per cent productivity gain for the tracked tasks. Eighty per cent of these tasks fell into the Requirements / Design phase of development, 20 per cent fell into the Construction phase. The tasks that were tracked for Architects in the Requirements/Design phase represented 35 per cent of all tasks a typical Architect would engage in during that phase. The overall gain for the Architect group for this phase, normalized to the complete life cycle, becomes 5.62 per cent (71.73*0.28*0.80*0.35).

Role		Architect			Business Analyst			Programmer Analyst			Total
Survey Scope Productivity Savings			71.73%			66.67%			60.68%		
	WBS	Task Distribution	Task Coverage	Overall Gain	Task Distribution	Task Coverage	Overall Gain	Task Distribution	Task Coverage	Overall Gain	
Requirements / Design	28%	80.00%	35.00%	**5.62%**	100.00%	35.00%	**6.53%**	-	-	-	**12.16%**
Request Initiation, Requirements Definition, Functional Design											
Construction	30%	20.00%	15.00%	**0.65%**	-	-	-	100.00%	20.00%	**3.64%**	**4.29%**
Detail Design, Programming, Unit Testing											
Testing	30%	-	-	-	-	-	-	-	-	-	0.00%
System, Regression, Acceptance, Volume/Stress Testing											
Documentation	4%	-	-	-	-	-	-	-	-	-	0.00%
Release Documentation, User's Guide, Online Help											
Management	8%	-	-	-	-	-	-	-	-	-	0.00%
Coordinate, Track/Update, Report											
Other	n/a	-	-	-	-	-	-	-	-	-	0.00%
Post-Implementation, Change Control, QA											
		100.00%		**6.27%**	100.00%		**6.53%**	100.00%		**3.64%**	**16.44%**
										16.44%	

Figure 4: Extrapolating Survey Results

Following similar steps, we can calculate the normalized productivity gain for Business Analysts to be 6.53 per cent in this example. There is no productivity gain for Programmer Analysts in the Requirements/Design phase, as no tasks were tracked for them.

The normalized productivity gain for the Requirements / Design phase is therefore 12.16 per cent in the above example. The TOTAL normalized productivity gain for the group comprising of all architects, business analysts and programmer analysts, based on the measured survey, comes to 16.44 per cent.

From this figure, one can easily conclude the overall anticipated productivity (and consequently monetary) gain for the entire IT department. If the above group represents 65 per cent of all staff complement, the ultimate savings for the department should be approximately 10.7 per cent (16.44 * 65/100).

© March 2009

Conclusion

The productivity gain calculated with the above method – an assessment of reduced task cycle time - represents the direct impact of the APM investment, and provides a valid basis in determining the return on the investment or the payback period. Note, however, that this gain is just the minimum improvement one can expect. An investment in APM offers a productivity solution that, although difficult to calculate its tangible versus intangible benefits, will prove to be a valuable tool to CIO-s looking to increase turnaround times and accuracy in their IT departments.

For more information please visit www.infobal.com, or call 416-962-5235.

© March 2009