

90% Within 3 seconds

Average of 1.5 seconds



95% Within 5 seconds

Application Performance Optimization
as the way to make **cost savings** as
well as deliver better performance

Philip Mann – Macro 4
October 2009

90% Within 3 seconds

Average of 1.5 seconds

What do these have in common?

95% Within 5 seconds

**They are the ways we set, measure, and report
response time service levels**

but

**How well do they reflect the actual end user
experience; and user satisfaction levels?**

and

**What is really going on in the service level
setting and reporting process?**

Presentation Overview

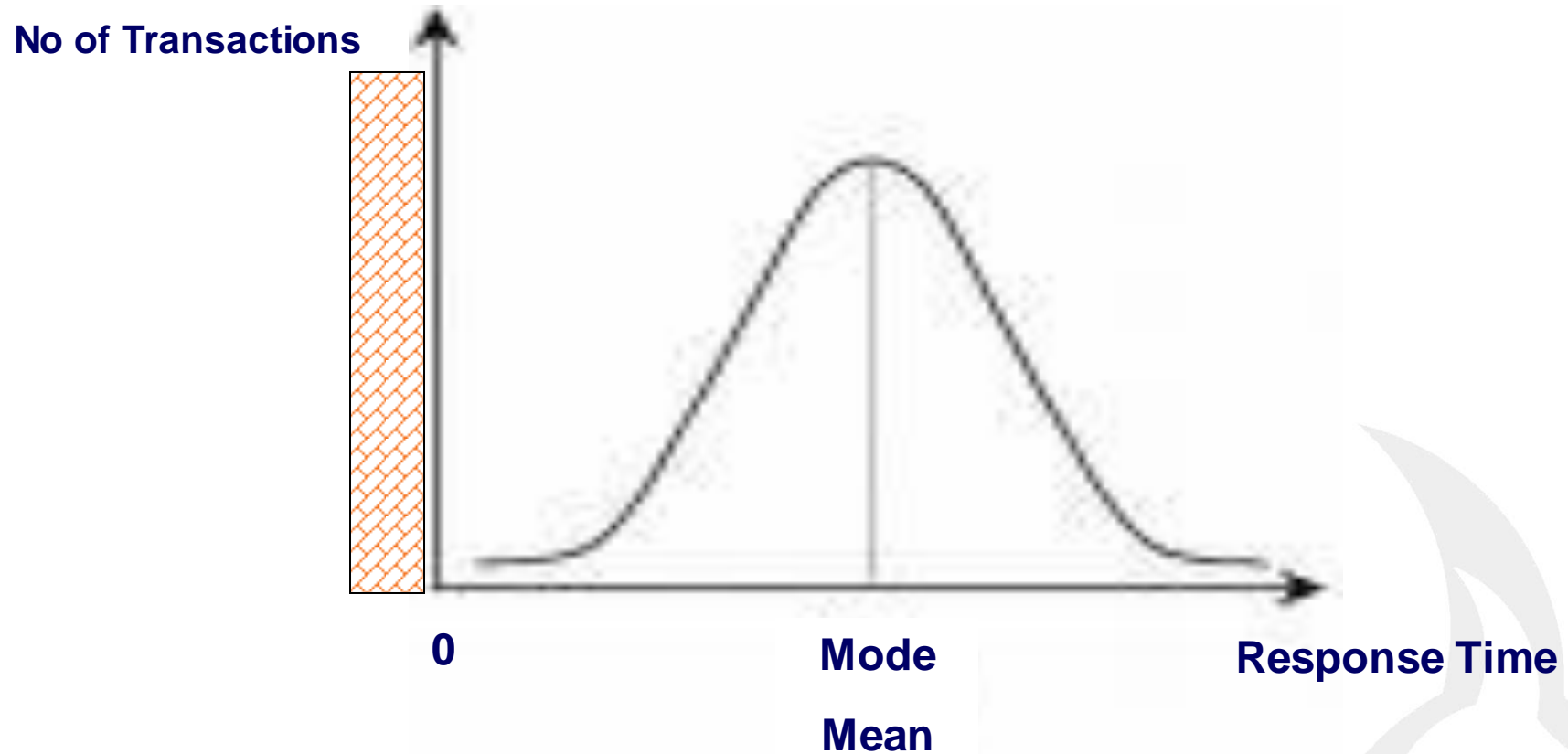
Look at how Service Management and other aspects of the Capacity Management process can be extended and made to pay handsome financial dividends

- What's needed to make this happen
 - begin to think a bit outside the box
 - and encourage new co-operative ways of working in our organisations
- What normally goes wrong
 - lack of ownership
 - rigid and inflexible organisational structures
- Introduce the Application Performance Optimization process
 - Case studies to illustrate what can really be achieved when barriers are broken down and 'Cash is King'

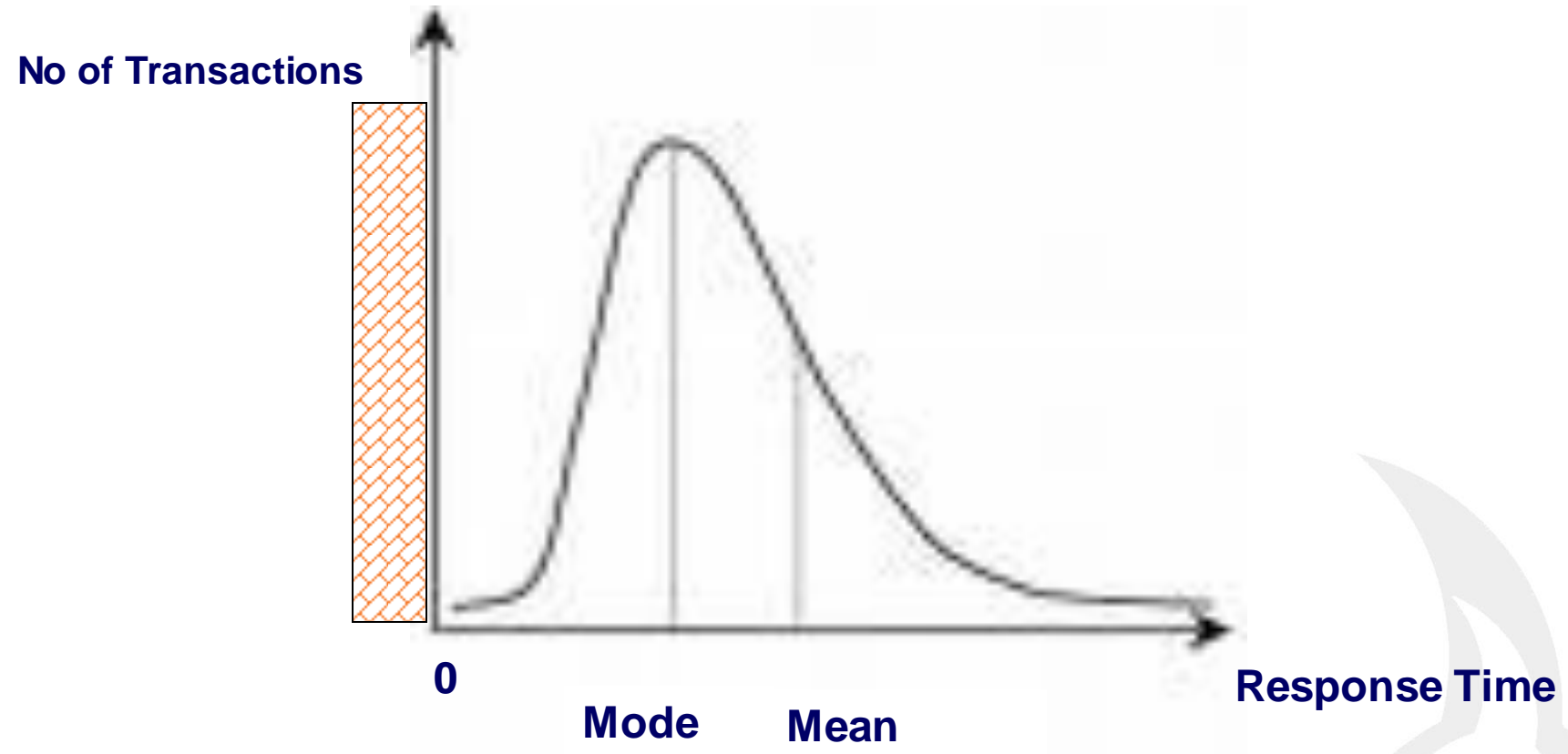
Service Management and Service Level Reporting

- The sort of things that we currently do
- Highlighting
 - Problems
 - Limitations
 - New opportunities

What do we think we mean by average?



What response times actually look like



What we do to combat this

- Response time statistics get more and more complicated
 - in an attempt to come up with something that better represents the end user experience
- Mean (average) response times are dropped
 - as they don't represent a response time that 'most' users experience
- Move to percentile reporting instead
 - 50% is some sort of average, but represents response that 50% don't get
 - 90%, 95% is better ...
 - but top few percent of 'rogue' transactions always have to be dropped
 - and don't count
- It gets more and more complicated
 - **And we're probably not measuring end-user response times anyway!**

If you are serious about reporting end user experience?

- Consider looking at APDEX



www.apdex.org

Service Management using Service Levels

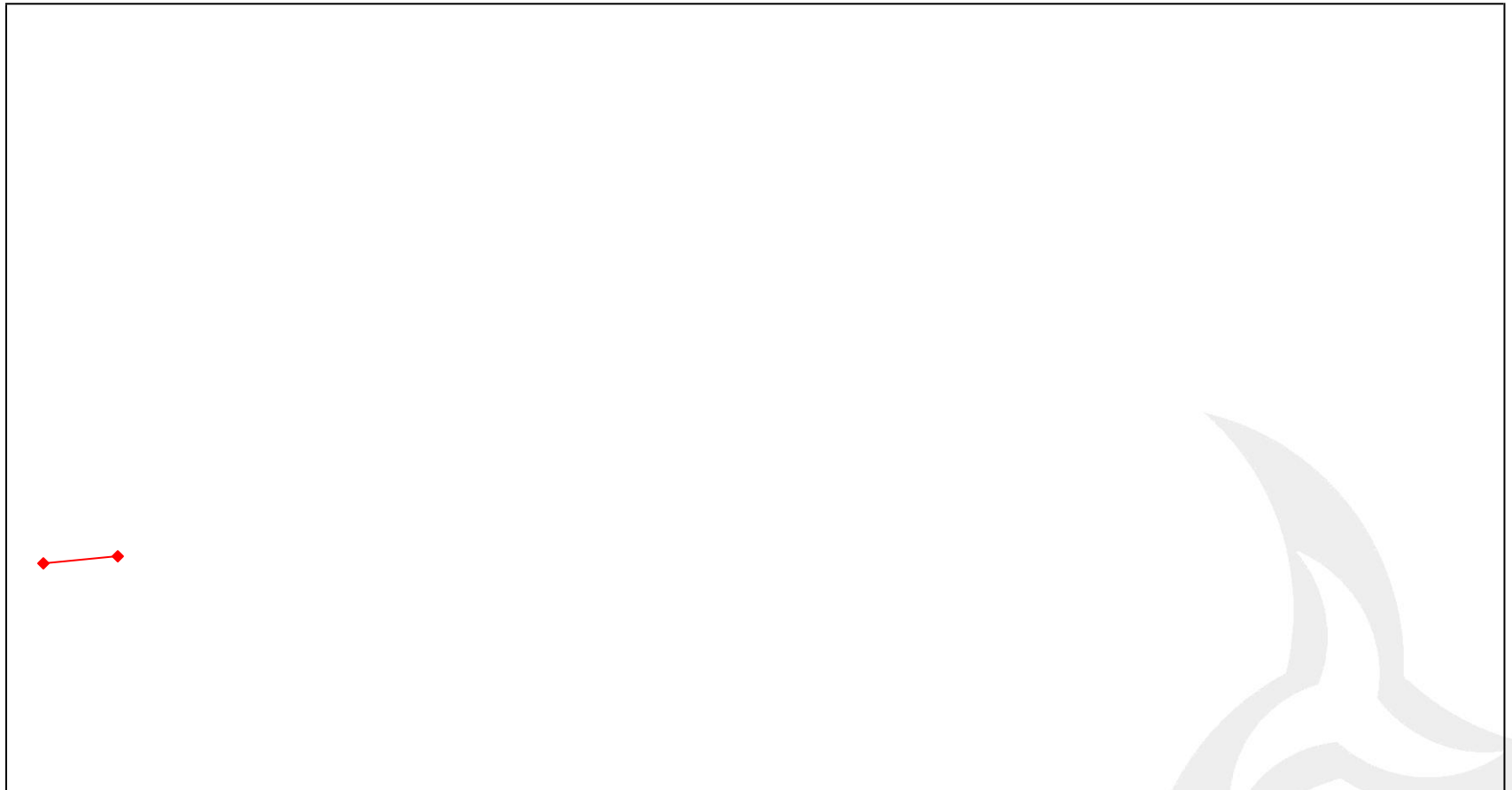
- It's fundamentally become a confrontational business (a game)
- Service Levels measure success or failure
 - They define the 'goalposts' in the game
- There are implied winners and losers
 - Service 'suppliers' are winners if service levels are met
 - Users are 'winners' if service levels are not met (via penalties that apply)
- There's not much other room for manoeuvre
 - Complaints not treated seriously if service levels 'OK'
 - Users insistent on keeping their service levels, even if they are impractical and even meaningless in commercial business terms

Graph of a Service Level Indicator (one measure)

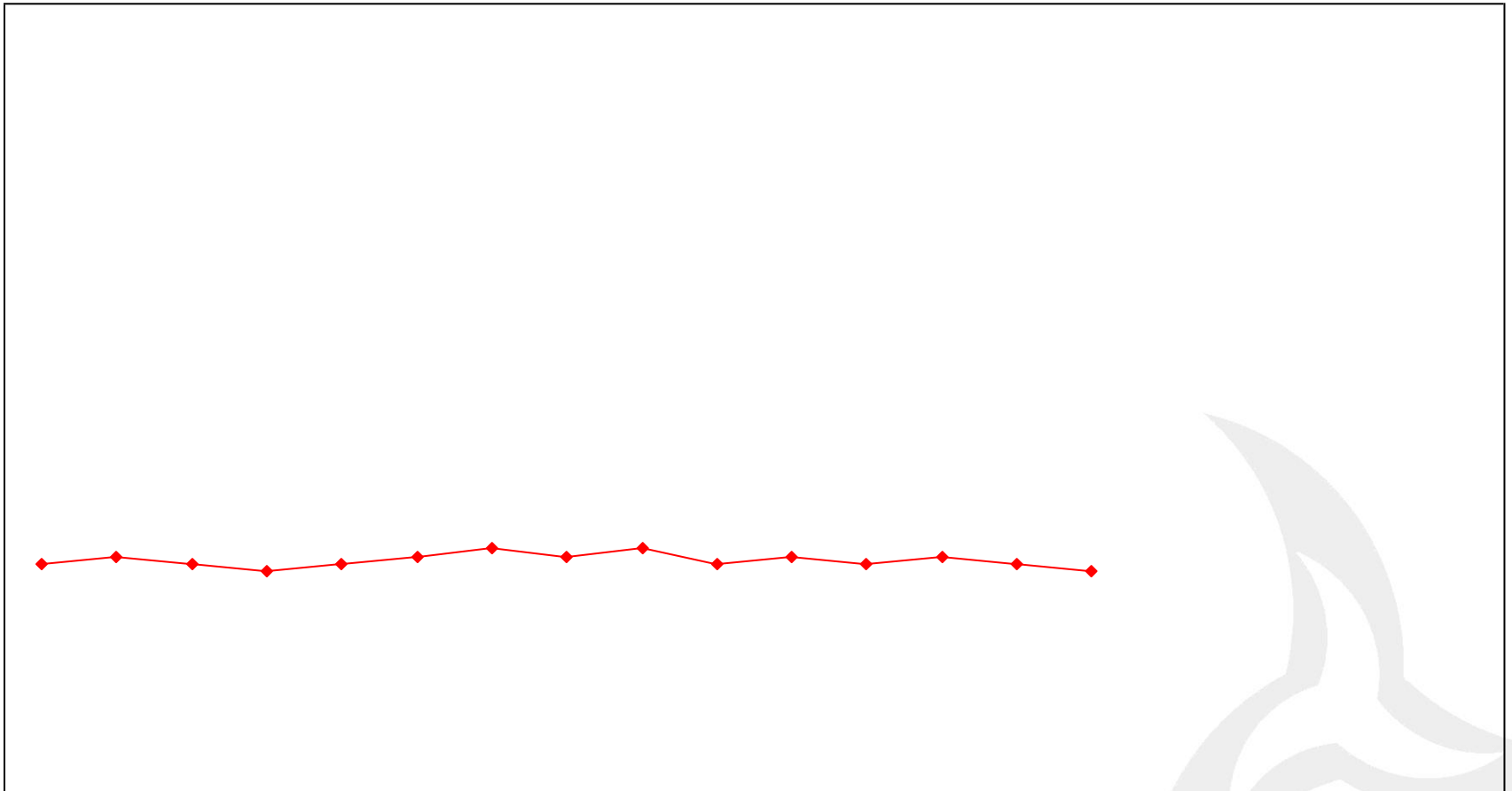
Introducing a better use of service level data



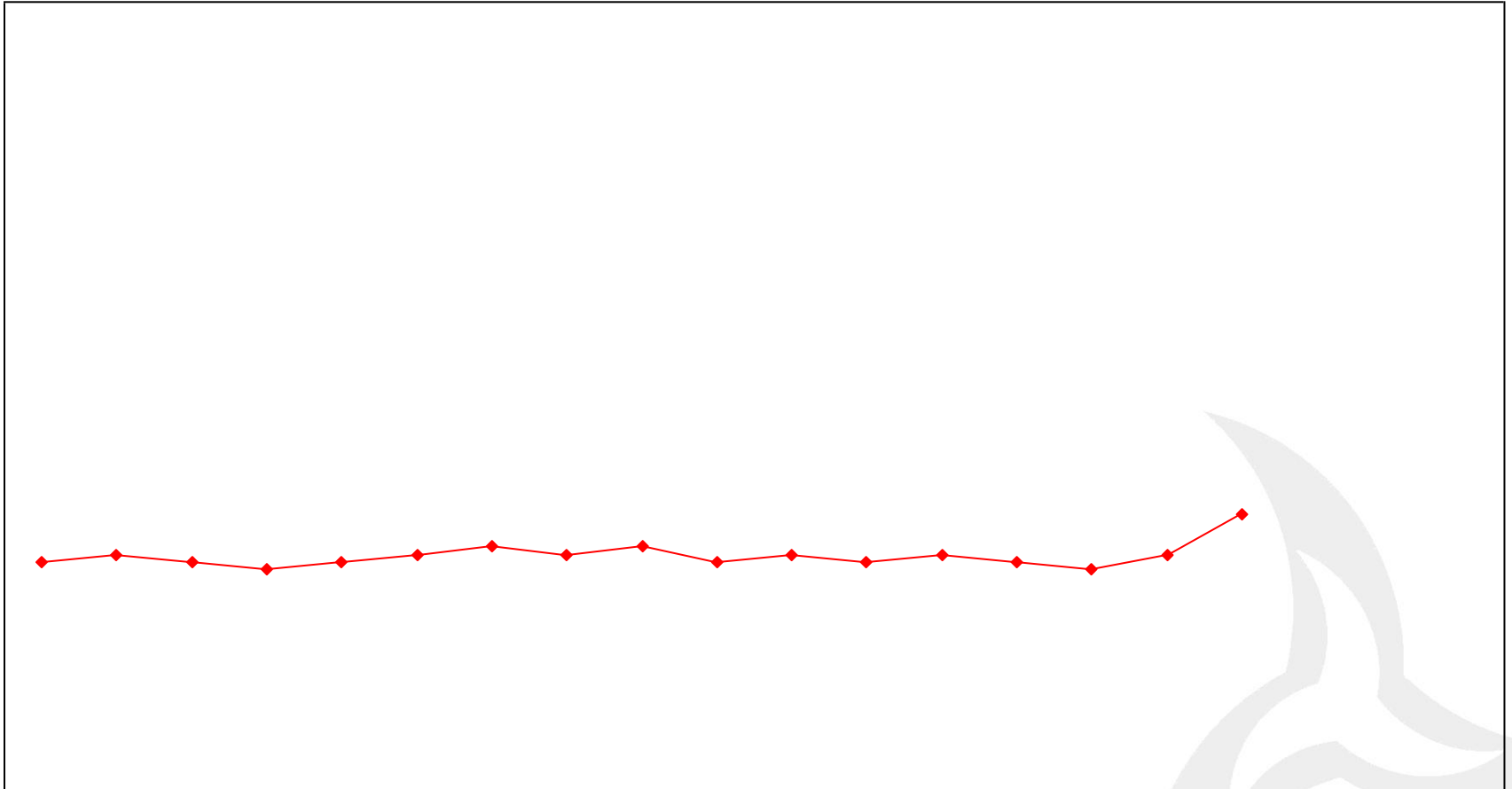
Graph of a Service Level Indicator (Trending)



Graph of a Service Level Indicator (Steady State)



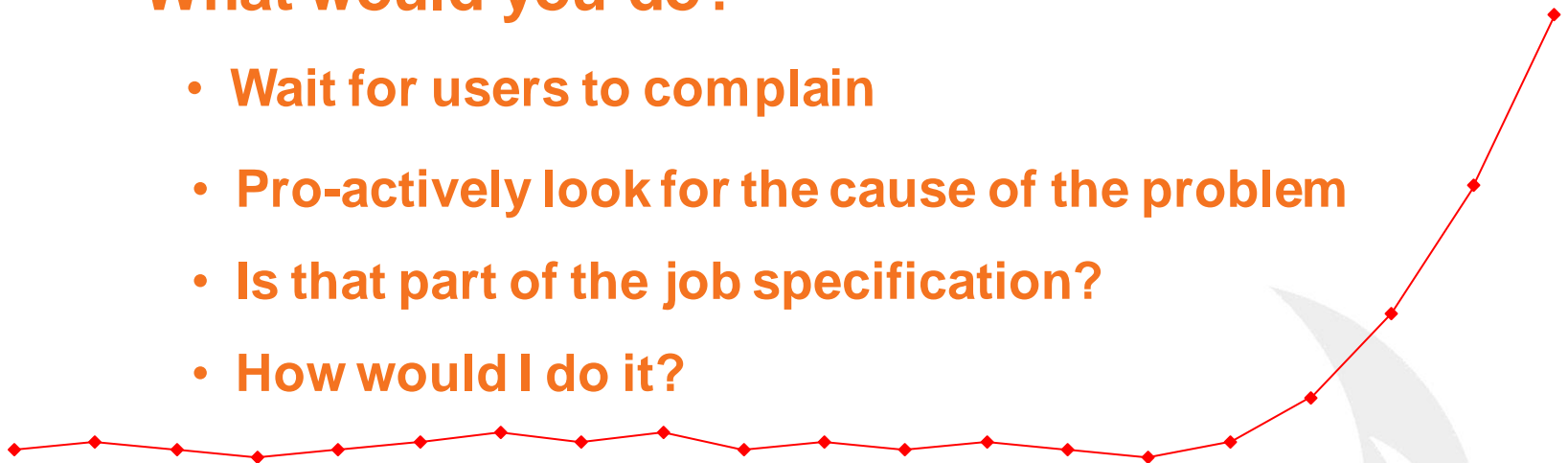
Graph of a Service Level Indicator (Cause for Concern?)



Graph of a Service Level Indicator (Definite Concern)

What would you do?

- Wait for users to complain
- Pro-actively look for the cause of the problem
- Is that part of the job specification?
- How would I do it?



Transactions

- We know what they are?
 - Components of company's computer systems – so they're important
 - Subject to Service Level Agreements and Service Level Reporting – we measure them
 - There may be penalties if 'we' get it wrong. Is this fair?
- Who owns them?
 - Ops/Production run them but do they 'own' them?
 - In fact they probably don't know a lot about what they do
 - Do development still own them? Where are the developers?
 - What about bought in packages?
- What happens when things go wrong?
 - Ops fix operational problems and some functional problems
 - Performance issues ignored unless service levels breached
 - No pro-active 'improvements'
- Lack of Ownership
 - **One of the major obstacles to continuous performance improvement**

Application Performance Optimisation

- What is it?
 - Process of looking at and improving the performance of the transactions that make up the applications that we run (and batch processes as well)
- What will it give us?
 - Improved service levels, response times, throughput etc.
 - Resource (CPU) usage reductions leading to cost savings
- Why should we do it?
 - If not for the service level improvements, then for the cost savings
 - But really, one comes with the other. It's a win, win situation
- Interested?

How to do it

- Start looking at things from a resource usage point of view
 - That's the money angle
 - The service improvements will follow
- Identify where resources are currently being used
 - SMF/RMF analysis
- Target application areas with high resource utilisation
- Work out what the applications actually do as they run
 - Via application profiling
- Identify areas of high resource utilisation **within** the applications
 - Target for tuning to reduce resource utilisation
 - And performance improvements will follow

Application Profiling

- The basis of the Application Performance Optimisation process
- Looks at things from the application rather than system point of view
- Gives a detailed and dynamic picture of what happens as application runs
 - Where and how the application uses CPU resources
 - And where it 'waits'
- Gets round application 'ownership', and 'expertise' problems
 - Because it doesn't matter anymore
 - You don't have to have prior knowledge or expertise
 - You can get your own (and better) picture of what is going on
- Provides focus
 - Identifies 'problems' which if fixed will deliver 'improvements'
 - Before and after observations can be used to prove savings

Example of an 'Application Profile'

- Taken with FreezeFrame
 - An application profiling tool
- Uses a sampling approach to observe the application as it runs
- Note the application orientated view

File View Navigate Help

C01: CPU Usage by Category (0074)

Row 00001 of 00003

Command ==>

Scroll ==> CSR

<u>Name</u>	<u>Description</u>	<u>Percent of CPU Time * 10.00%</u> ±1.1%
		* . . . 1 . . . 2 . . . 3 . . . 4 . . . 5 . . . 6 . . . 7 . . . 8
<u>DB2SQL</u>	SQL Processing	99.86
<u>SYSTEM</u>	System/OS Services	0.08
<u>APPLCN</u>	Application Code	0.05



File View Navigate Help

C01: CPU Usage by Category (0074)

Row 00001 of 00031

Command ==>

Scroll ==> CSR

Name	Description	Percent of CPU Time * 10.00% ±1.1%	
			* 1 2 3 4 5 6 7 8
DB2SQL	SQL Processing	99.86	
→ 00007	M4A377J(5015) OPEN	97.18	
→ 00013	M4A377J(5652) UPDATE	0.73	
→ 00008	M4A377J(4751) SELECT	0.22	
→ 00017	M4A377J(5936) OPEN	0.22	
→ 00009	M4A377J(5727) SELECT	0.19	
→ 00011	M4A377J(4272) SELECT	0.19	
→ 00015	M4A377J(5434) FETCH	0.13	
→ 00010	M4A377J(5814) SELECT	0.13	
→ 00018	M4A377J(5254) FETCH	0.12	
→ 00014	M4A377J(5057) FETCH	0.09	
→ 00012	M4A377J(4118) INSERT	0.09	
→ 00004	M4CJJ2J(14073) FETCH	0.08	
→ 00029	M4A377J(5979) FETCH	0.06	
→ 00030	M4A377J(6043) CLOSE	0.06	
→ 00019	M45GB7A(2414) UPDATE	0.04	
→ 00022	M45GB7A(1981) SELECT	0.04	

File View Navigate Help

F11: SQL CPU/Service Time by Statement (0074)

Row 00001 of 00035

Command ==>

Scroll ==> CSR

SeqNo	Name	stmt#	SQL Function	Nbr of SQL Calls	--CPU Total	Time-- Mean	--Svc Total	Time-- Mean
S0007	M4A377J	5015	OPEN	53	360.33	6.79871	1510.69	28.50374
S0013	M4A377J	5652	UPDATE	5,812	2.33	0.00040	23.26	0.00400
S0008	M4A377J	4751	SELECT	7,489	1.03	0.00013	31.92	0.00426
S0017	M4A377J	5936	OPEN	1,727	0.77	0.00045	12.77	0.00739
S0009	M4A377J	5727	SELECT	4,502	0.69	0.00015	25.51	0.00566
S0012	M4A377J	4118	INSERT	2,968	0.53	0.00018	2.47	0.00083
S0011	M4A377J	4272	SELECT	5,793	0.46	0.00007	1.41	0.00024
S0010	M4A377J	5814	SELECT	4,502	0.37	0.00008	13.55	0.00301
S0015	M4A377J	5434	FETCH	2,977	0.36	0.00012	15.62	0.00525
S0023	M4A377J	4596	UPDATE	2,825	0.32	0.00011	0.67	0.00023
S0014	M4A377J	5057	FETCH	5,865	0.32	0.00005	0.62	0.00010
S0004	M4CJJ2J	14073	FETCH	4	0.26	0.06556	6.86	1.71639
S0018	M4A377J	5254	FETCH	2,977	0.25	0.00008	1.98	0.00066
S0021	M4A377J	5552	SELECT	871	0.15	0.00017	0.31	0.00036
S0029	M4A377J	5979	FETCH	1,727	0.09	0.00005	0.22	0.00013
S0027	M45GB7A	2213	UPDATE	342	0.08	0.00026	0.17	0.00052

What performance problems can be found

- Anything from simple application problems to more complex system ones
- Particularly productive in finding high resource utilising DB2/SQL calls
- Other areas include
 - Excessive use of INITIALISATION code in app programs
 - Performance problems in common routines
 - Use of built in functions (LE routines)
 - And many more

Case Studies

- Large UK Supermarket Chain
 - Wanted to break out of annual cycle of CPU upgrades
 - Avoided upgrade for 2008 Christmas Peak period
 - Saved in excess of 15% of MIPS previously in use
 - Have now trained up their own staff to make further big savings
- UK Water Authority
 - Wanted to save a specific amount of processing power (100 MIPS)
 - Commissioned a study (which found potential for more than 100MIPS)
 - Paid for consultants to implement the savings that they required

So how do I go about saving my company money?

- DIY - Do it yourself
 - buy the required products for application profiling
 - ExpeTune, FreezeFrame and ExpeTune/DB
 - get extensive, practical training, delivered it in your environment
 - and possibly some consultancy to get you going
- Consultancy Based
 - Usually involves short engagement to quantify potential savings
 - Various options then for:
 - acquiring necessary products
 - realising the savings that have been identified
 - **Tends to breakdown barriers and cut across problems that stop the DIY approach from working**



Thank You

Philip.Mann@macro4.com