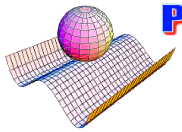
Capacity Planning Boot Camp

Part II: Metrics and Management

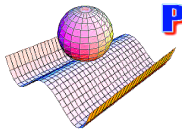
Dr. Neil J. Gunther
Performance Dynamics Company
Castro Valley, California
www.perfdynamics.com

CMG-T Session 415
CMG 2008 Conference, Las Vegas, Nevada

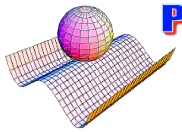


Performance Dynamics Company℠

Table of Contents



Performance Metrics



Some Standard Performance Metrics

Symbol	Metric	Definition
T	Measurement period	Same as the sample period
W	Waiting time	Time spent in a buffer
S	Service time	Time spent getting processed
B	Busy time	Total time the resource is busy
C	Completion count	Number of completed requests
R	Residence time	Time spent waiting and being serviced
$\mathcal{R}$	Response time	Sum of all the residence times
X	Throughput (C/T)	Rate at which work is completed
ρ	Utilization (B/T)	Fraction of T the resource is busy
Q	Queue length	Total number of requests in the system

Some Fundamental Metric Relations

$$X = \frac{C}{T} \tag{1}$$

$$\rho = \frac{B}{T} \tag{2}$$

$$S = \frac{B}{C} \tag{3}$$

$$R = W + S \tag{4}$$

Little's Law Means a Lot

Comes in 2 flavors:

1. Little's macroscopic law:

$$Q = XR \tag{5}$$

2. Little's microscopic law:

$$\rho = XS \tag{6}$$

Basic idea:

$$\#\text{miles} = \frac{\text{miles}}{\text{hour}} \times \text{hours}$$

The metrics on the LHS are both pure numbers (no units):
Q is the number of requests in *residence* (R).
ρ is the number of requests in *service* (S, no waiting) .

Metrics in UNIX

UNIX is an experiment that escaped from the lab in 1975 and has been mutating ever since.

There is no such thing as UNIX: Name three?

UNIX included memory locations called *counters* that are the same counters used by every performance management product today.

They were never intended to provide system performance analysis or capacity planning information. (cf. RMF and SMF)

So why are they there?

iostat Command in Linux

Consider the number of disk IOs reported in `iostat`[a].

```
iostat [ interval [ count ] ]
```

The *interval* argument specifies the reporting period or sample period in seconds. A *count* parameter can be specified in conjunction with the interval parameter to control how many reports are generated before iostat exits.

The first report always displays information since the system was booted, while each subsequent report covers the time period since the last report.

```
avg-cpu:  %user   %nice    %sys   %idle
           6.11    2.56    2.15   89.18

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
dev3-0            1.68        15.69        22.42   31175836   44543290
```

The single disk (`dev3-0`) has averaged 1.68 transfers (IOs) per second.

[a] Assumes Sysstat Suite is installed.

How It Works

Suppose the sample period or interval is $T = 30$ seconds.

And the count is $C = 50$ IOs during that interval.

The IO rate is calculated as:

$$\frac{C}{T} = \frac{50}{30} = 1.67 \text{ TPS} \tag{7}$$

This is the same thing as IO throughput.

Throughput is the number of completed requests per unit time. ($X = C/T$ from slide 4)

mpstat Command in Linux

Now consider the `mpstat`[a] command.

```
07:09:26 PM  CPU   %user   %nice %system   %idle    intr/s
07:09:26 PM  all    6.40    5.84    3.29   84.47    542.47
```

On this system there are actually two CPUs as shown by issuing `mpstat -P ALL` instead.

```
07:13:03 PM  CPU   %user   %nice %system   %idle    intr/s
07:13:03 PM  all    6.40    5.84    3.29   84.47    542.47
07:13:03 PM    0    6.36    5.80    3.29   84.54    542.47
07:13:03 PM    1    6.43    5.87    3.29   84.40    542.47
```

The average CPU utilization is: $100\% - \%\texttt{idle} = 15.53\%$.

Note this is the same as: `%user + %nice + %system`.

[a] Assumes Sysstat Suite is installed.

How It Works

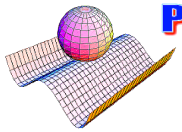
Suppose the sample period or interval is $T = 30$ seconds.

And the CPU busy time is $B = 4.66$ seconds.

The CPU utilization is calculated as:

$$\frac{B}{T} = \frac{4.66}{30} = 0.1553 \tag{8}$$

Recall $\rho = B/T$ from slide 4.



Simple Bottleneck Analysis

Little's Law and Bottlenecks

Recall Little's micro-law (slide 6):

$$\rho = XS \tag{9}$$

If the resource reaches saturation then $\rho = 1$ (or 100% busy).

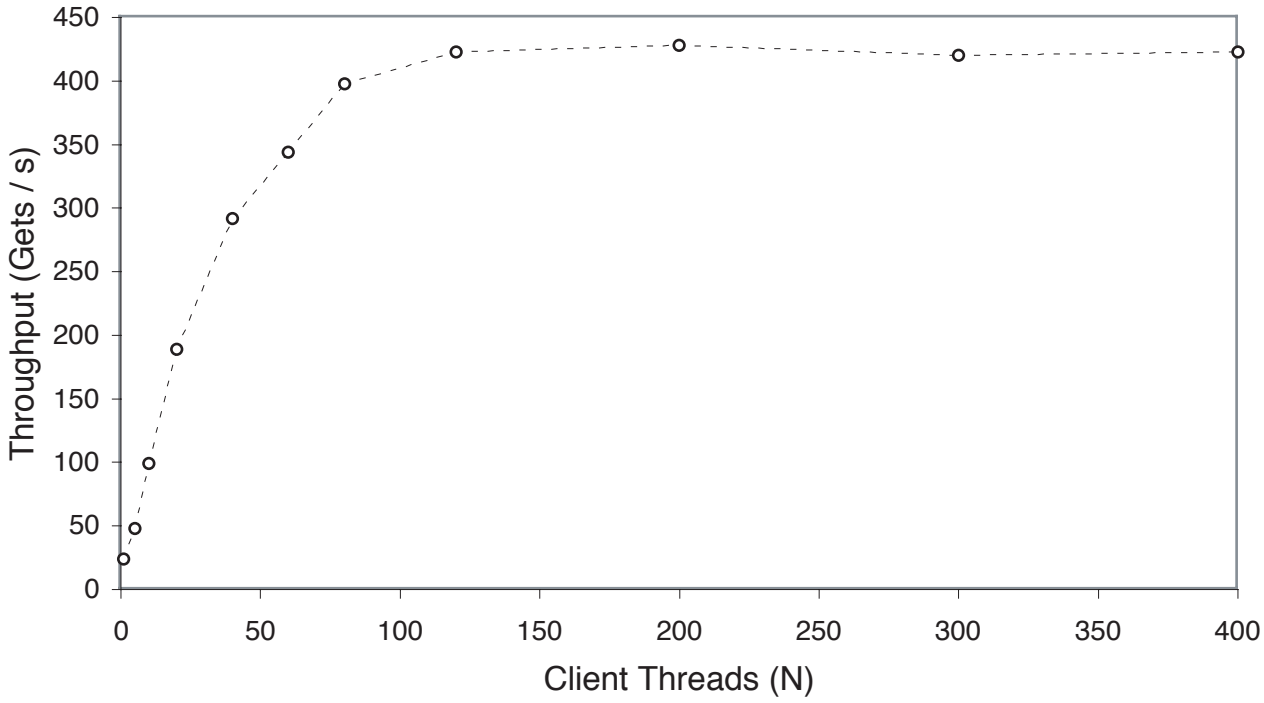
$$1 = XS \tag{10}$$

$$X = \frac{1}{S} \tag{11}$$

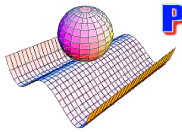
The bigger the value of S the lower the value of X.

The resource with $S_{max} = max(S_1, S_2, \ldots, S_k)$ is the (primary) bottleneck.

Typical Throughput Characteristic



S_{max} controls the height of the plateau in the throughput.



A Fiasco in Florida

Pyramid joint project for an insurance company in Florida c.1992.

Involved a mainframe running CICS being replaced by a Pyramid 24-way server.

Pyramid VP asked me to look at the performance engineering results.

I call the insurance company test engineers in Florida.

They tell me their best throughput is 50 TPS measured by RTE on the mainframe with an average think time set to $Z = 30$ seconds.

Internal instrumentation showed something like: $22, 30, \ldots, 10$ ms.

To which I responded:

> *"If the instrumentation data is not broken, then the best your throughput can be is only 33 TPS."*

Florida performance engineers (who had been working on this project for 18 months 7×24) were flabbergasted!

Data Comes from the Devil

Guerrilla maxim 1.13: *Busy work does not the truth maketh.* If you don't take time off to come up for air and reflect on what you're doing, how are you going to know when you're wrong?

That night (while I went home to sleep), the Florida test engineers spent another sleepless night trying to accomplish one thing ...

Models Come from God

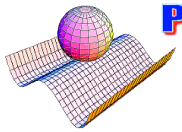
The longest (bottleneck) process takes 30 ms.

Therefore, that process determines the highest possible throughput.

$$X_{max} = 1/S_{max} = \frac{1}{0.030} \tag{12}$$

That is, 1 tx every 30 ms or 33 tx's every second = 33 TPS.
(See slide 24 about significant digits)

To achieve 50 TPS, the bottleneck process would have be 20 ms.



Riddle Solved

The intended $Z = 30$ seconds was not being asserted by the RTE.

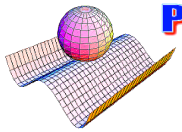
It was being treated as $Z = 0$ (default or batch).

The intense arrival rate into the SUT meant some txs were not being processed.

But were being accounted for by the RTE as successfully completed.

The measurement was really: 50 TPS = 33 TPS + 17 errors/sec

Guerrilla maxim 2.12: *If the measurements don't support your performance predictions, change the measurements.*



Sins of Precision

Sources of Error

All measurements contain errors.

1. Systematic - same amount in each repeated measurement
2. Random - differing amount in each repeated measurement

A simple ROT (see Part I about Rules-Of-Thumb) for instrumentation error is:

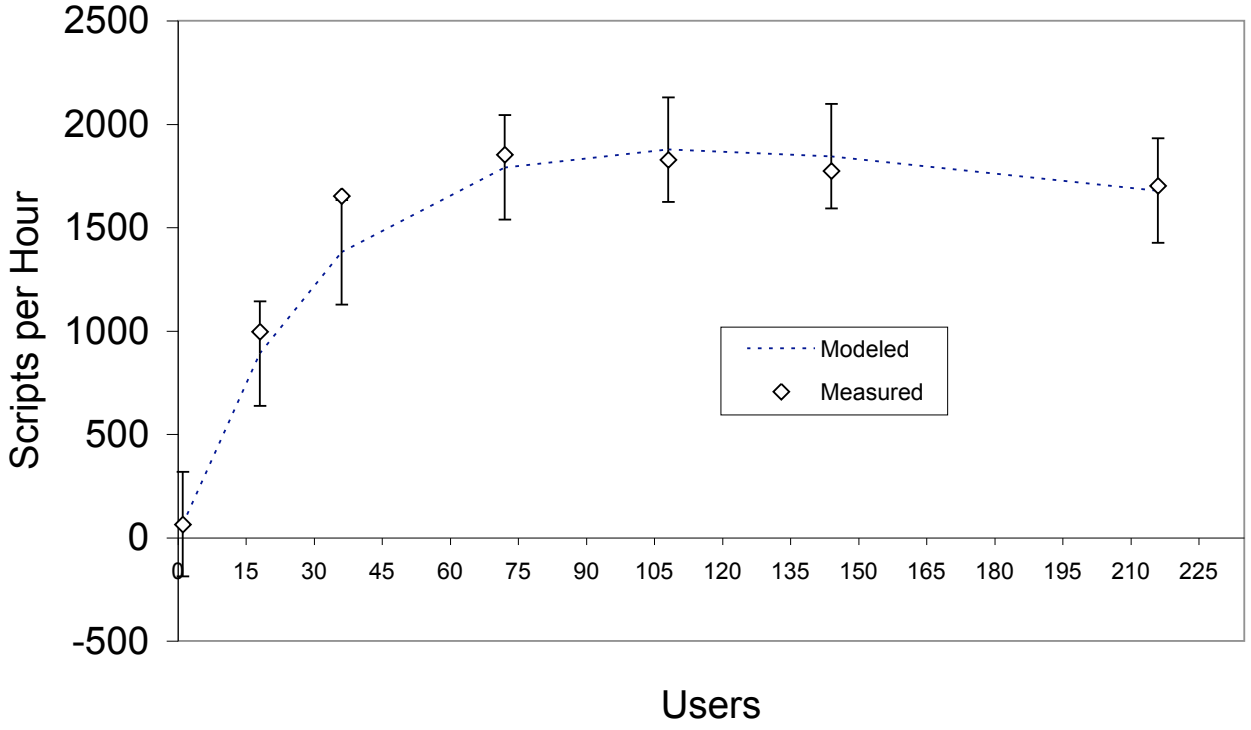
$$X \pm \Delta X \tag{13}$$

where ΔX is taken as $1/2$ the smallest measurement unit of the instrument.

Example: In UNIX and Linux the smallest time resolution for sampled measurements is 1 second. Therefore: $\Delta T = \pm 0.5$ seconds.

Another approach is to calculate the *confidence interval* (CI). Requires repeated measurements to determine CI. See Ch. 4 of [3].

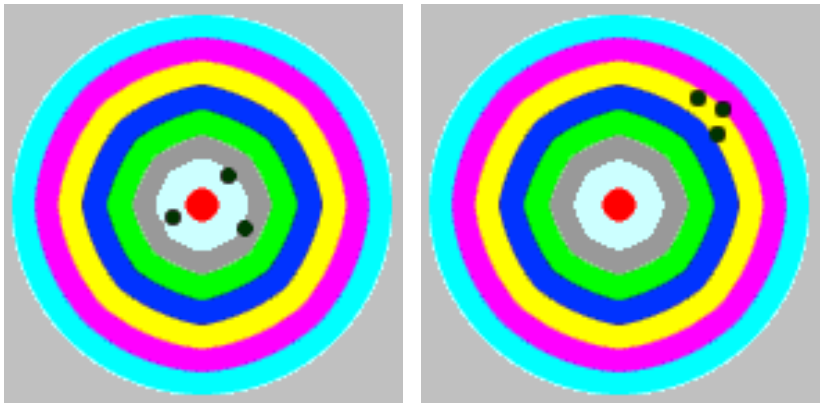
Error Bars



All plots should have error bars.

CaP predictions (e.g., models) are only required to pass through all the error bars, not all the data points.

Accuracy vs. Precision



Suppose the "true" value is represented by the bull's eye.

Accuracy: the spread of points about the bull's eye.

Precision: the spread of points with respect to each other.

Therefore, you can achieve high precision with low accuracy.

Precision and Significant Digits

Question 1: Is there any difference between these numbers?:

$$3, 3.0, 3.00, 3.000$$

Question 2: Is there any difference between these numbers?:

$$50, 0.005, 0.00005$$

Answer to Question 1

The precision of a measurement is expressed by the number of *significant digits* (hereafter "SigDigs"). Each of the numbers $3, 3.0, 3.00, 3.000$ has the same magnitude but different precision.

How to Determine SigDigs:

```
Always scan Left to Right
Is there an explicit decimal point?

YES: Locate the first non-zero digit
     Count it and ALL digits (including zeros) to its right

 NO: Insert a decimal point on the end
     Locate the last non-zero digit prior to the decimal point
     Count it and ALL digits to its left
     Ignore all zeros trailing that digit
```

Answer 1. Number of SigDigs: $1, 2, 3, 4$.

Answer to Question 2

Question 2: $50, 0.005, 0.00005$

Answer 2. Number of SigDigs: $1, 1, 1$.

In this case, the *accuracy* is different but the *precision* (number of significant digits) is the same.

Although may seem odd at first, think of it this way:

$$0.005 \text{ seconds} \equiv 5 \text{ milliseconds}$$

In other words, the number of sigdigs is the same (1), only the measurement units have changed.

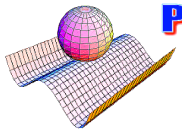
Historical Screw Up

Same precision, different accuracy:

$$\overbrace{3.\overbrace{141592654}^{accuracy}}^{precision}$$

$$\overbrace{3.\overbrace{142000000}^{accuracy}}^{precision}$$

In 1853, π was calculated to about 700 places. In 1949 a computer found an error starting near the 500th place. (See [2])



Some More Examples

Number	SigDigs	Remark
50	1	Implicit decimal point
0.00341	3	
1.0040	5	
50.0005	6	
6.751	4	
0.157	3	
28.0	3	
40300	3	Implicit decimal point
0.070	2	
30.07	4	
65000	2	Implicit decimal point
0.0067	2	
6.02×10^{23}	3	Explicit decimal point in scientific notation

Rounding Rule

The following is the most up to date version of the rounding algorithm [2].

Suppose the terminating string of digits is: $\ldots\ X\ Y\ Z$

a. Examine Y

b. If $Y < 5$ then goto (i)

c. If $Y > 5$ then set $X = X + 1$ and goto (i)

d. If $Y \equiv 5$ then examine Z

e. If $Z \geq 1$ then set $Y = Y + 1$ and goto (a)

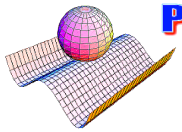
f. If Z is blank or a string of zeros then

g. Examine the parity of X

h. If X is odd then set $X = X + 1$

i. Drop Y and all trailing digits

See www.perfdynamics.com/Tools/tools.html for VBA, Mathematica and Perl versions of this rounding algorithm.



Multiplication and Addition

Multiplication: When multiplying or dividing measurements, the answer should be rounded down to the same number of SigDigs as the measurement with the *least* SigDigs.

Addition: *Before* adding or subtracting measurements, round them to the same degree of precision as the measurement with the least SigDigs.

CPU Utilization Revisted

Recall the example on slide 11.

Sample period $T = 30$ sec and CPU busy time $B = 4.66$ sec. The CPU utilization was calculated by division:

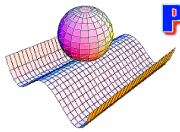
$$\frac{B}{T} = \frac{4.66}{30} = 0.1553$$

But there are 4 SigDigs, whereas 30 only has 1 SigDig.

Apply the Rounding Rule to: $X\ Y\ Z \equiv 1\ 5\ 5$.
Since $Y \equiv 5$, we examine $Z \equiv 5$.
Since $Z > 1$, we increment $Y \rightarrow Y \equiv 6$ and re-examine it.
Since $Y > 5$ now, we increment $X \rightarrow X \equiv 2$.
Finally, drop Y and all trailing digits to produce:

$$\frac{B}{T} = \frac{4.66}{30} = 0.2$$

which is correct to 2 SigDigs.



ITIL and Capacity Management

What is ITIL?

ITIL: *Information Technology Infrastructure Library.*

Comprises a set of (expensive) manuals (the "library" part) which define a framework for IT within the context of business processes.

ITIL was initiated in the 1980's by British *Office of Government Commerce* (OGC) as a set of best practices for IT Service Management (ITSM), and OGC still owns the copyright.

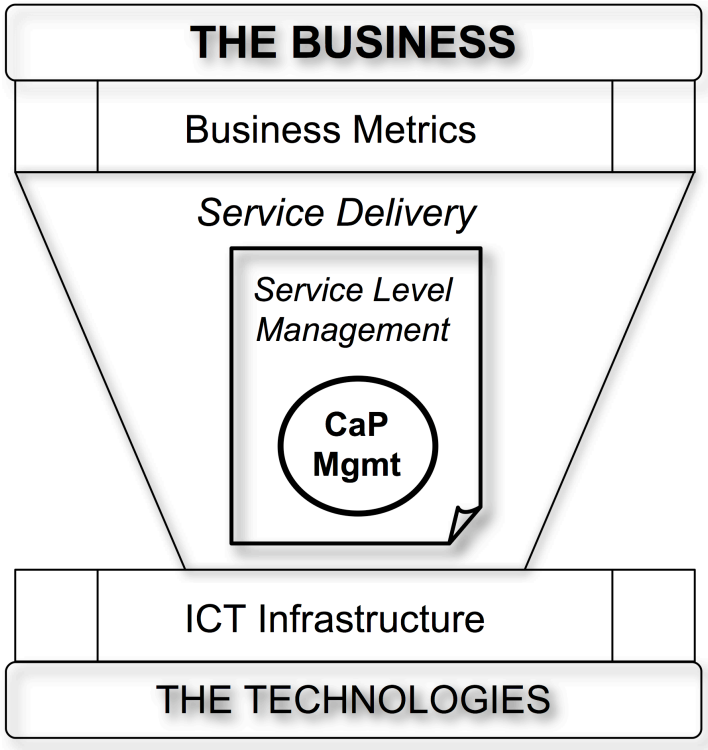
Possibly dues to this chauvinism, it has been slow to take off in the USA [5] .

It has a presence at CMG, and this year I counted 9 scheduled presentations on ITIL.

ITIL is about *process*, not *implementation.*

ITIL is about *what*, not *how.*

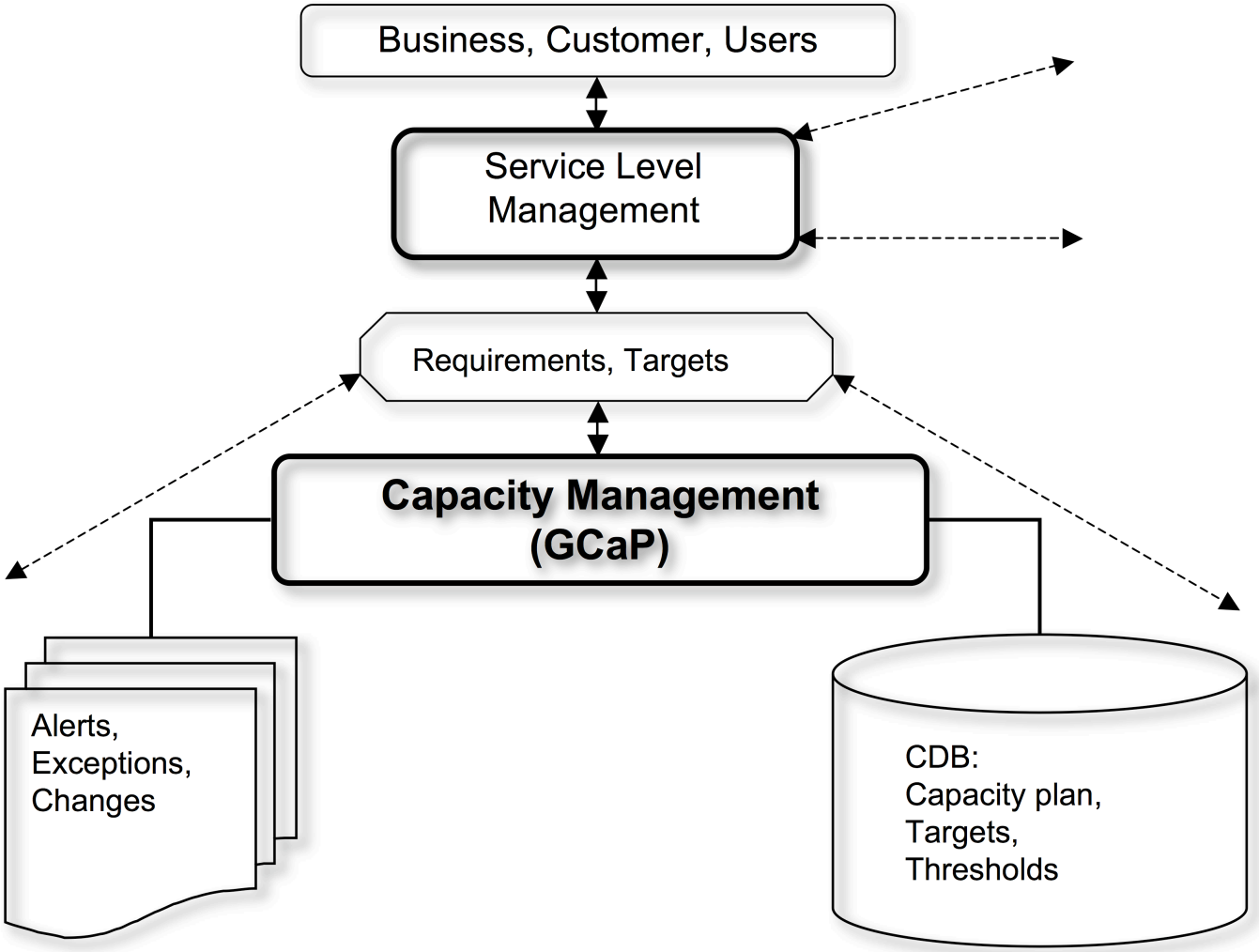
ITIL Framework

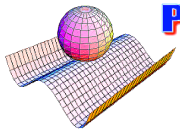


ITIL buries capacity management down in the bowels of *Service Delivery* and *Service Level Management*.

A managed coupling between IT technologies and business metrics.

ITIL CaP Management





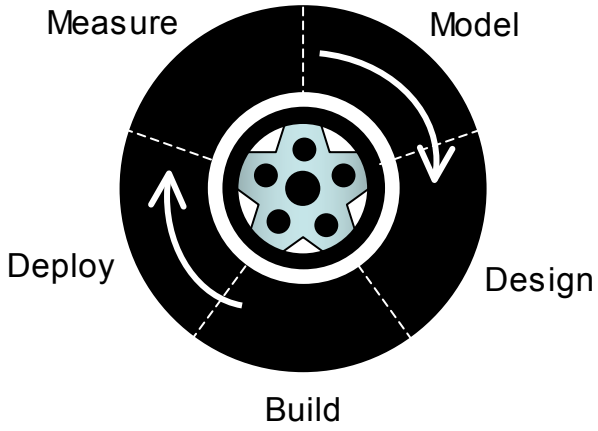
Performance Dynamics Company℠

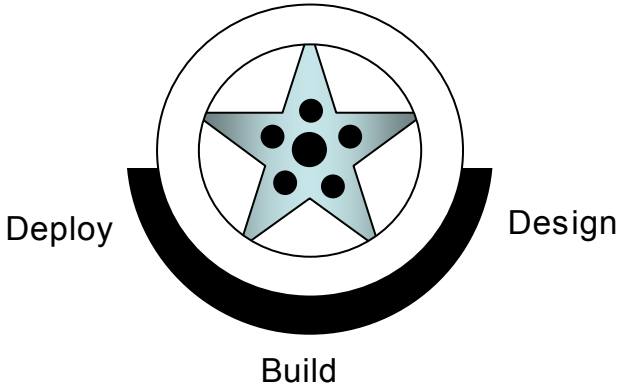
A Role for GCaP

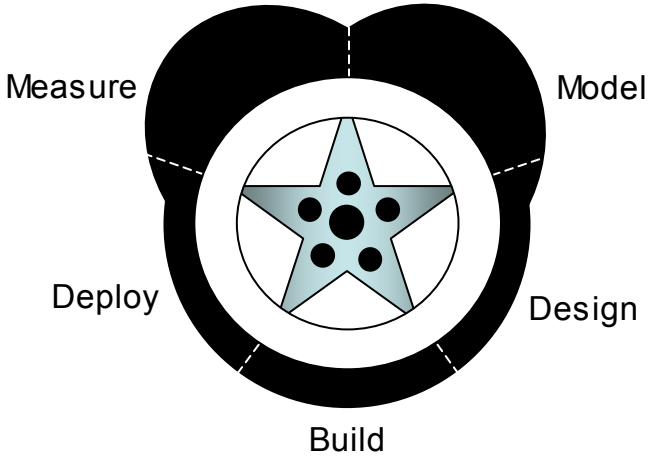
Since implementation is undefined in ITIL, GCaP should fit the bill by definition. ☺

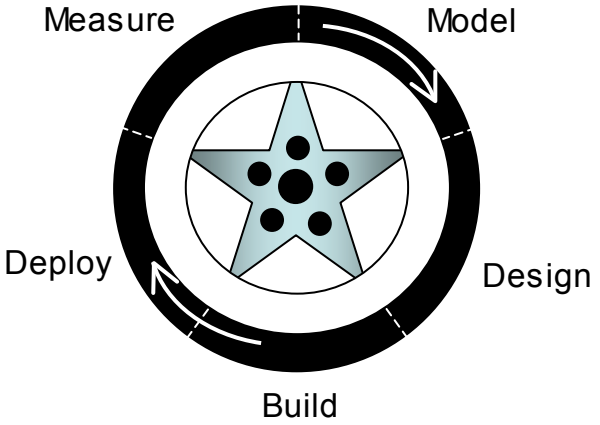
Attribute	Gorilla	Guerrilla
Planning	Strategic	Tactical
Horizon	12 to 24 months	3 mins to 3 months
Focus	Hardware	Applications
Budget	Not an issue	None that you know of
Title	On your office door	Not even on your business card
Style	Opulent and ponderous	Lean and mean
Tools	Expensive commercial	Commercial and open-source
Reporting	Routine, written, formal	Opportunistic, verbal, informal
Skills	Specialized, more quantitative	Eclectic, more qualitative

Guerrilla Wheel of Capacity Planning









Summary for Part II

Should now understand the definitions and relationships between some of the most commonly used metrics in performance analysis and capacity planning.

These relationships can provide powerful tools for capacity planning e.g., bottleneck analysis.

All performance measurements come with errors. They are usually not presented that way, but you have been warned.

Data comes from the Devil. Trust but verify. Measurement precision is contained in the number of significant digits. You can't have more than was measured.

CaP needs to couple to business processes.

ITIL is the latest framework in which this can be done.

GCaP is a plausible implementation of CaP within ITIL.

References

[1] N. J. Gunther, *Analyzing Computer System Performance with Perl::PDQ*, Springer, 2005

[2] N. J. Gunther, *Guerrilla Capacity Planning*, Springer, 2007

[3] D. J. Lilja, *Measuring Computer Performance*, Cambridge, U.P., 2000

[4] N. J. Gunther, CMG 2002, Reno, Nevada, pp. 433–446.

[5] *it*SMF, IT Service Management Forum, The USA chapter is at http://www.itsmfusa.org/.

