Performance Testing:

Essential Information for the Entire Team

Created for:

By:

Scott Barber

Chief Technologist

PerfTestPlus, Inc.

Performance Testing:

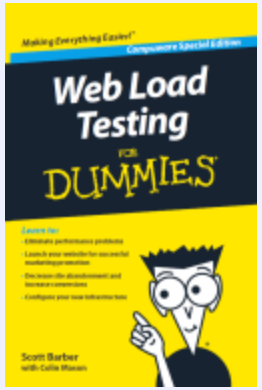
Essential Information for the Entire Team

Scott Barber

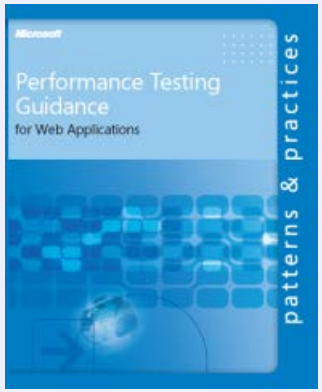
Chief Technologist, PerfTestPlus, Inc.
sbarber@perftestplus.com
www.perftestplus.com
@sbarber

Co-Founder: Workshop On Performance and Reliability
www.performance-workshop.org

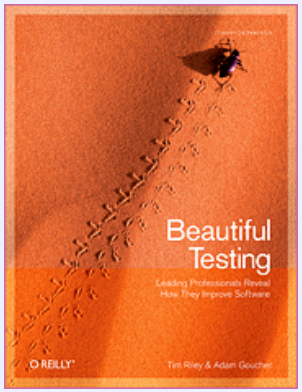
Author:

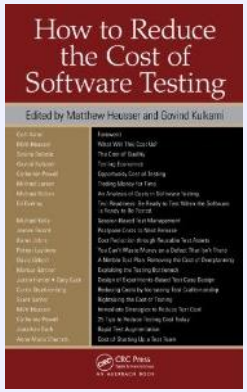


Co-Author:



Contributing Author:





Books: www.perftestplus.com/pubs

About me: about.me/scott.barber

WARNING

I speak 1 language (not very "naturally")

I speak quickly (especially when I'm telling stories)

I choose words very carefully (that may or may not translate well)

This class is a compilation of 7 days of instruction (about the same as a 1 semester university course)

I do not believe in "Best Practices" (I believe in solving problems via experience, experiment & education)

I have success using/doing everything in this presentation in *some* context (but not yours)

Some suggestions are harsh (use at your own risk)

I *like* to be challenged & interact with you

To Get The Most From This Class

Ask questions **(helping you understand is very important to me)**

Adapt concepts to your situation/context **(you can ask me for help with that)**

Don't worry about what "the boss" will or won't allow **(at least not before you I finish explaining)**

Do laugh at my jokes **(or groan – so I know if it didn't translate well)**

Do network with others who are here **(I bet they have great ideas too)**

Do not be shy **(did I mention that I *like* to be challenged and interact with you?)**

Follow–up with me **(when something works for you… or doesn't)**

Fact:

One *does not need* to be a performance testing rock star to have a significant positive impact on performance…

…and thus add significant business-value…

…quickly and simply.

A few more minutes?

Discuss with the people near you:

What is “Performance”?

What is “Performance Testing”?

Who is responsible for Testing Performance?

Consider making some notes:

If you have a connected device, you may take notes at http://typewith.me/p/NEXT2012

Put your initials (e.g. [rsb]) in front of your notes

I will ask a few volunteers to share their thoughts

"Let's face the truth, performance testing

****IS* rocket science."***

--*Dawn Haynes*

… but even rocket science involves

****SOME* easy stuff.***

--Addendum added by: *Scott Barber*

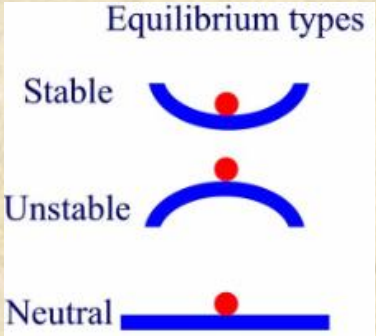
What is Performance?

System or application characteristics related to:

Speed:
- responsiveness
- user experience

Scalability:
- capacity
- load
- volume

Stability
- consistency
- reliability
- stress



What is Performance Testing?

What mom tells people:

I help people make websites go fast.

What I tell people:

I help and/or teach individuals and organizations to *optimize software systems* by balancing:

- Cost
- Time to market
- Capacity

while remaining focused on the *quality of service to system users.*

Performance vs. Load Testing?

Performance is to Load

as

Rectangle is to Square

Fact:

As an activity, performance testing is widely misunderstood, particularly by executives and managers.

This misunderstanding can lead to a variety of difficulties -- including *outright project failure.*

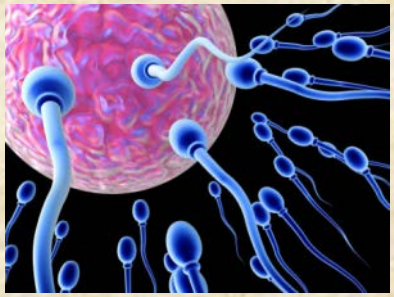
Fact:

Managers and executives *do not* need to understand the technical details of performance testing to make good decisions or effectively manage performance testing projects.

They *do* need to understand what performance testing is, what it is not and what value it adds.

The Performance Lifecycle is:

Conception to Headstone

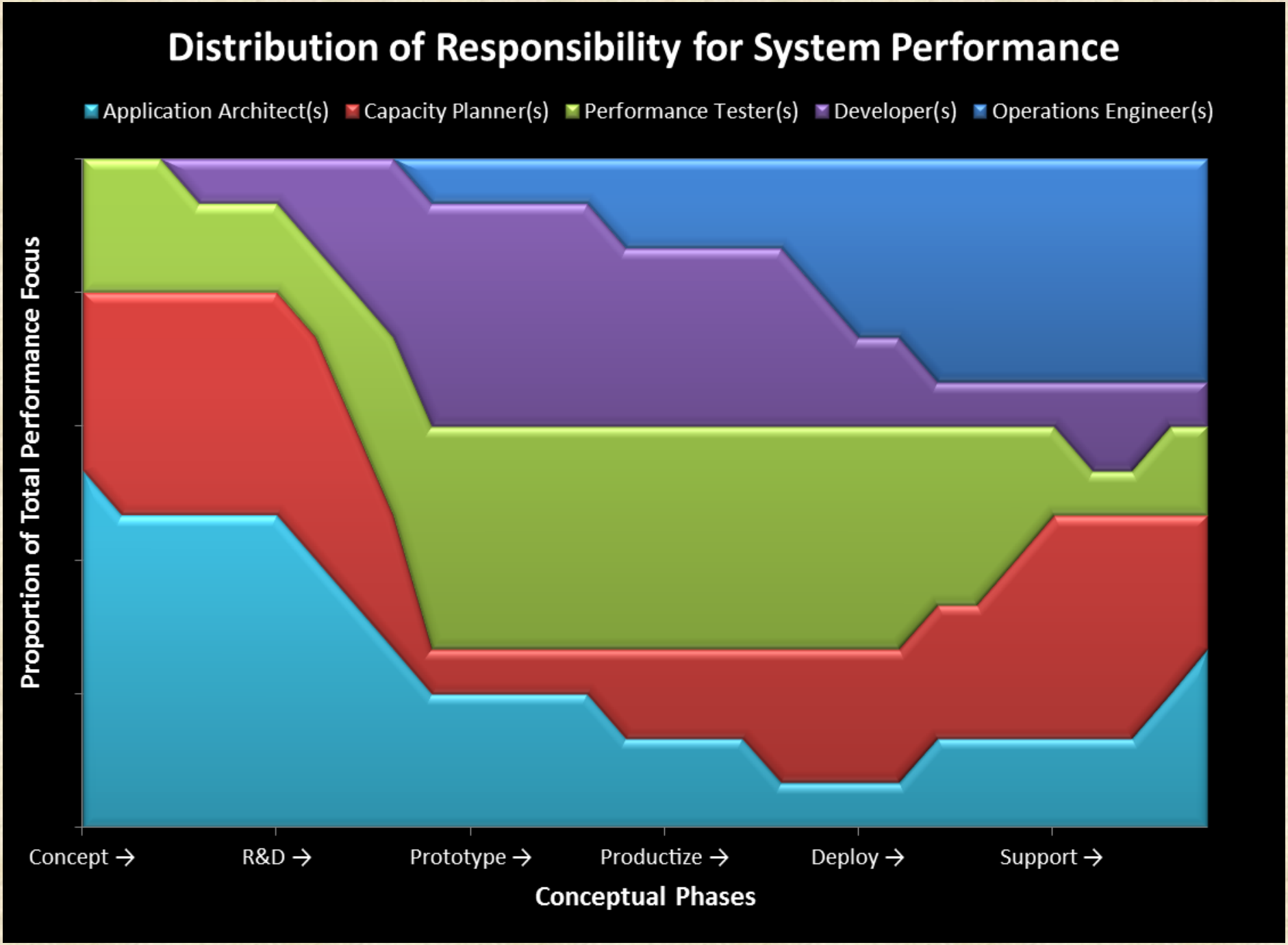


Not

Cradle to Grave

Who is Responsible?

Everyone



Let's Review:

System/Application Performance relates to… ?

Speed, Scalability, Stability

Performance Testing is… ?

Testing Focused on Speed/Scalability/Stability

Load Testing is… ?

A small subset of Performance Testing

Executives & Managers… ? (be nice)

Have no clue… and don't need much of one.

The Performance Lifecycle is…?

At least as long as the application's lifecycle.

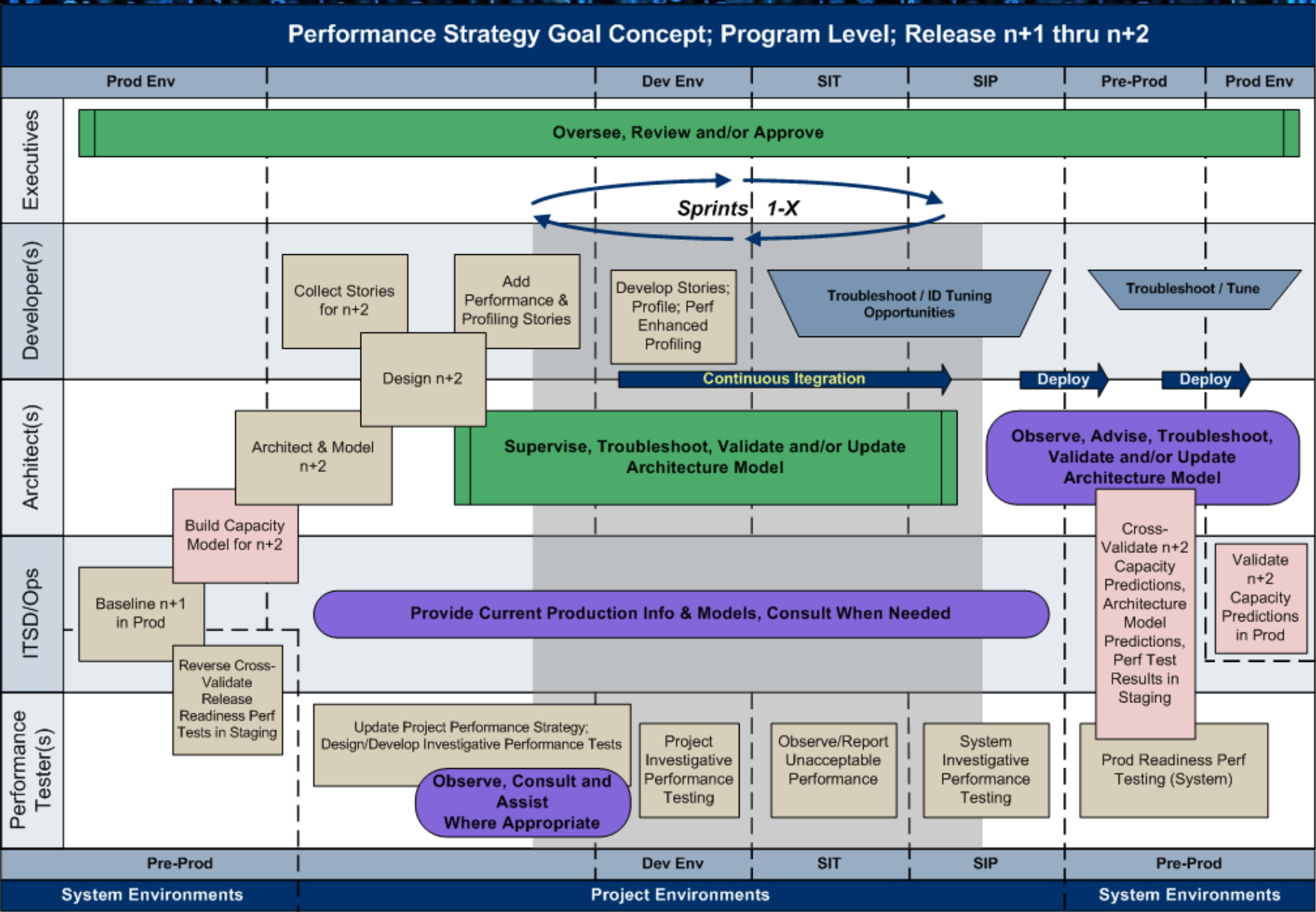
Who is responsible for performance?

Pretty much everyone.

Prevent Poor Performance…

… don’t just react when it happens.

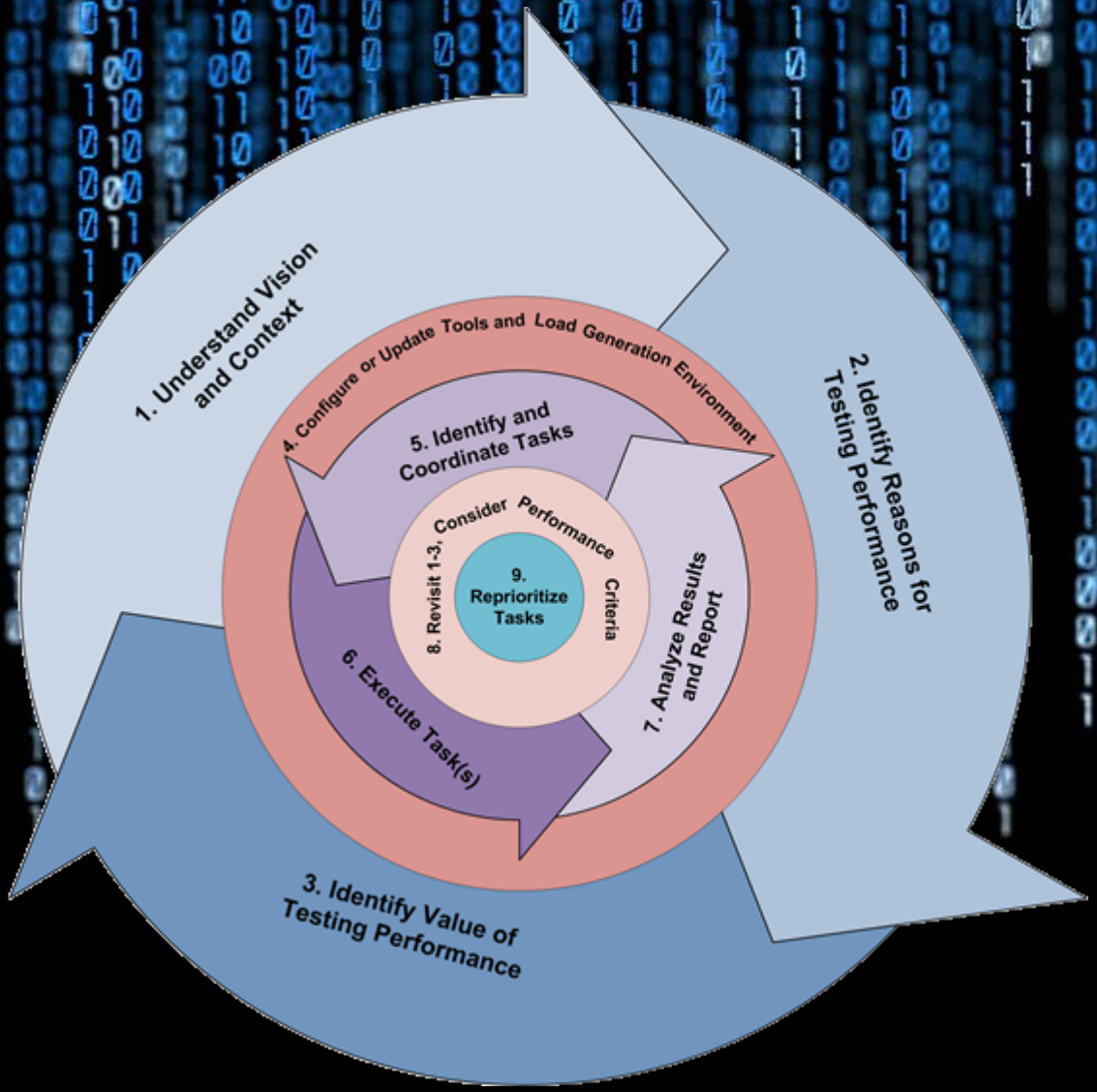
… But where does it fit in the SDLC?



...Or maybe like this?

Agile Performance Testing Activities

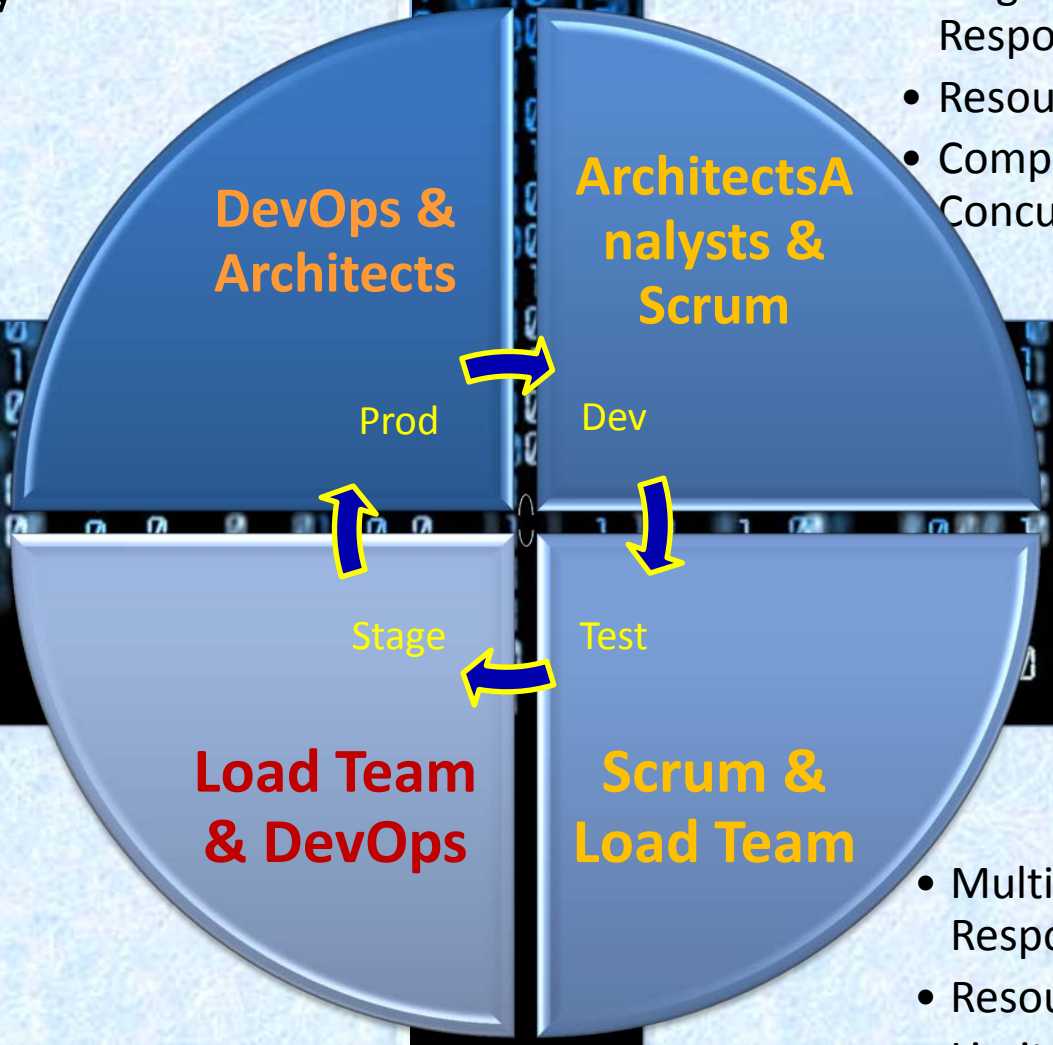
1. Understand Project Vision and Context
2. Identify Reasons for Testing Performance
3. Identify Value of Testing Performance
4. Configure or Update Tools and Load Generation Environment
5. Identify and Coordinate Tasks
6. Execute Task(s)
7. Analyze Results and Report
8. Revisit 1-3, Consider Performance Criteria
9. Reprioritize Tasks



...Or this?

- Current Capacity
- Projections
- Scalability Plan
- Reliability

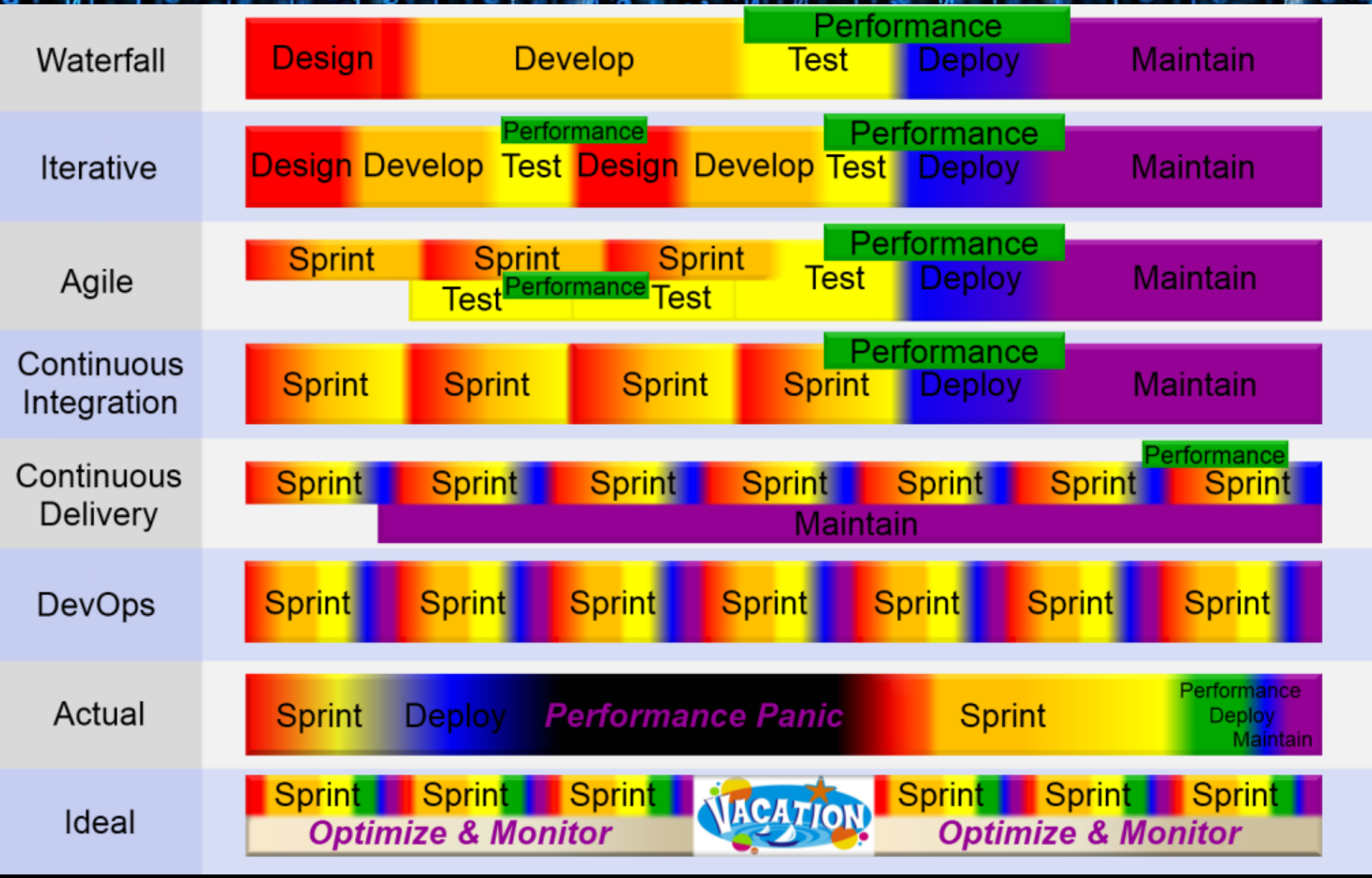
- Single User Responsiveness
- Resource Management
- Component-Level Concurrency



- Prepare for Prod
- Tune/Optimize
- Predict
- Early Warning Identification

- Multi-User Responsiveness
- Resource Trending
- Limit Identification

...Or maybe even?



… But *I* prefer…

Develop

(Sprint, Iteration, etc.)

Goals/Budgets

Unit Tests/Trends

Story Acceptance

Build Validation/Regression

Integrate

(Build, Merge, Promote, etc.)

Features

Target Load Profiles

Peak Load Profiles

Weakness Identification

Production

Deployments

Maintenance

Rapid Response

Data Forward

Prepare

(Major Release, Configure, Certify, etc.)

External Integrations

Infrastructure/Tuning

Break Point Identification

Capacity Planning


Develop
(Sprint, Iteration, etc.)
Goals/Budgets
Unit Tests/Trends
Story Acceptance
Build Validation/Regression
Integrate
(Build, Merge, Promote, etc.)
Features
Target Load Profiles
Peak Load Profiles
Weakness Identification
Target
Tune
Test
Production
Deployments
Maintenance
Rapid Response
Data Forward
Prepare
(Major Release, Configure, Certify, etc.)
External Integrations
Infrastructure/Tuning
Break Point Identification
Capacity Planning
Trend

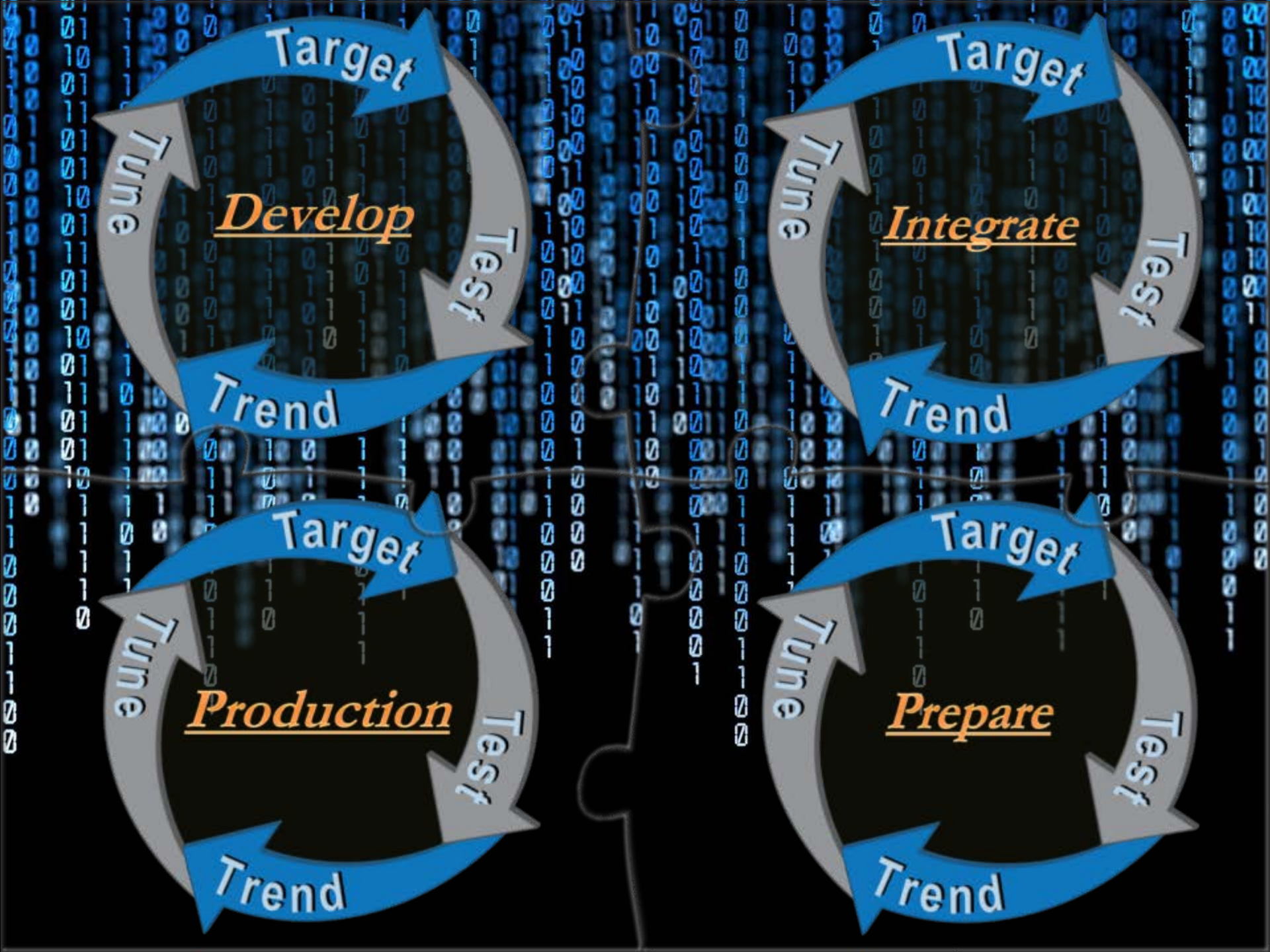
A *"Test-Driven Performance"* Model

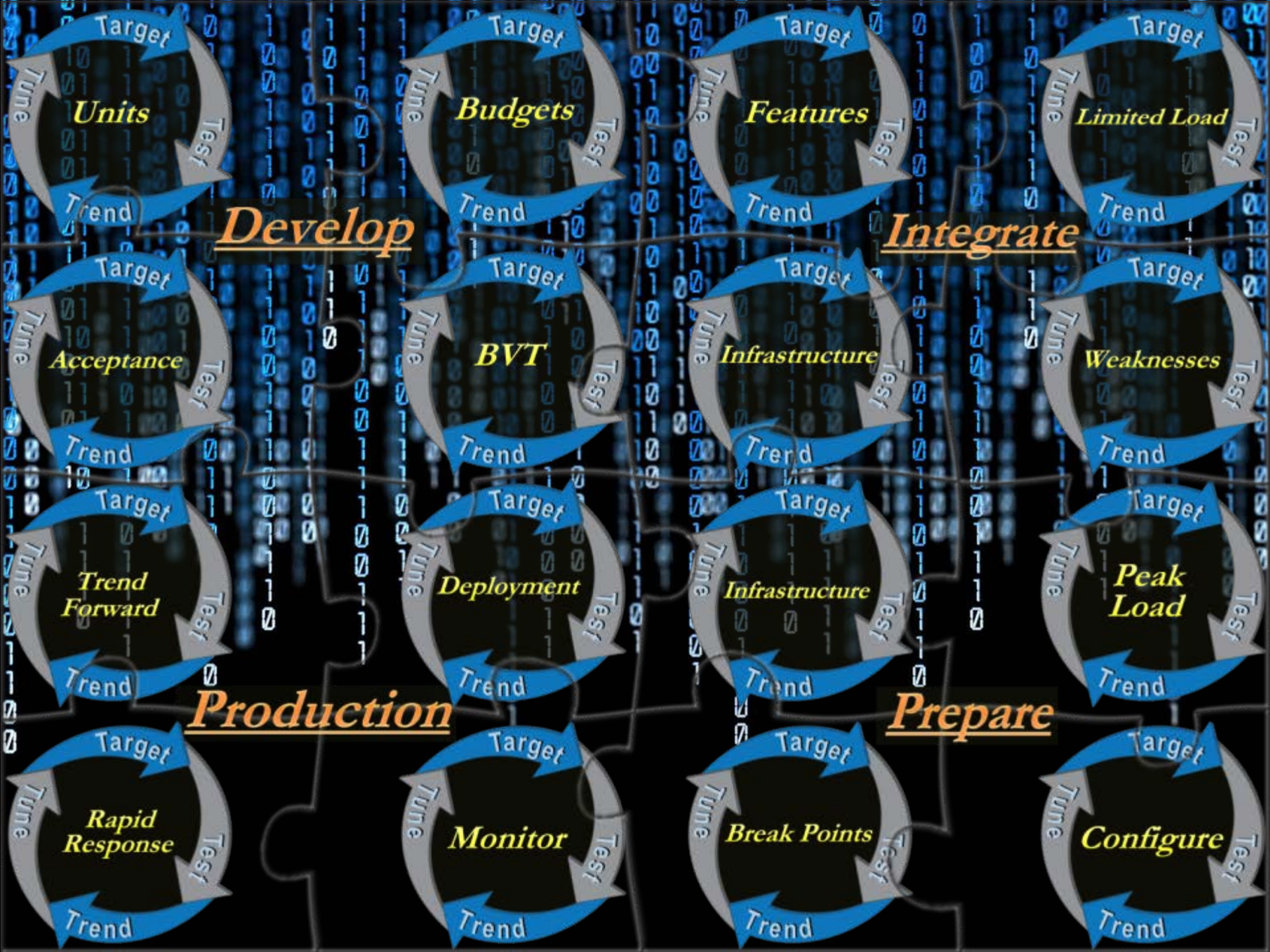
Proactive
Micro & Macro
Establish Goals
Update Targets

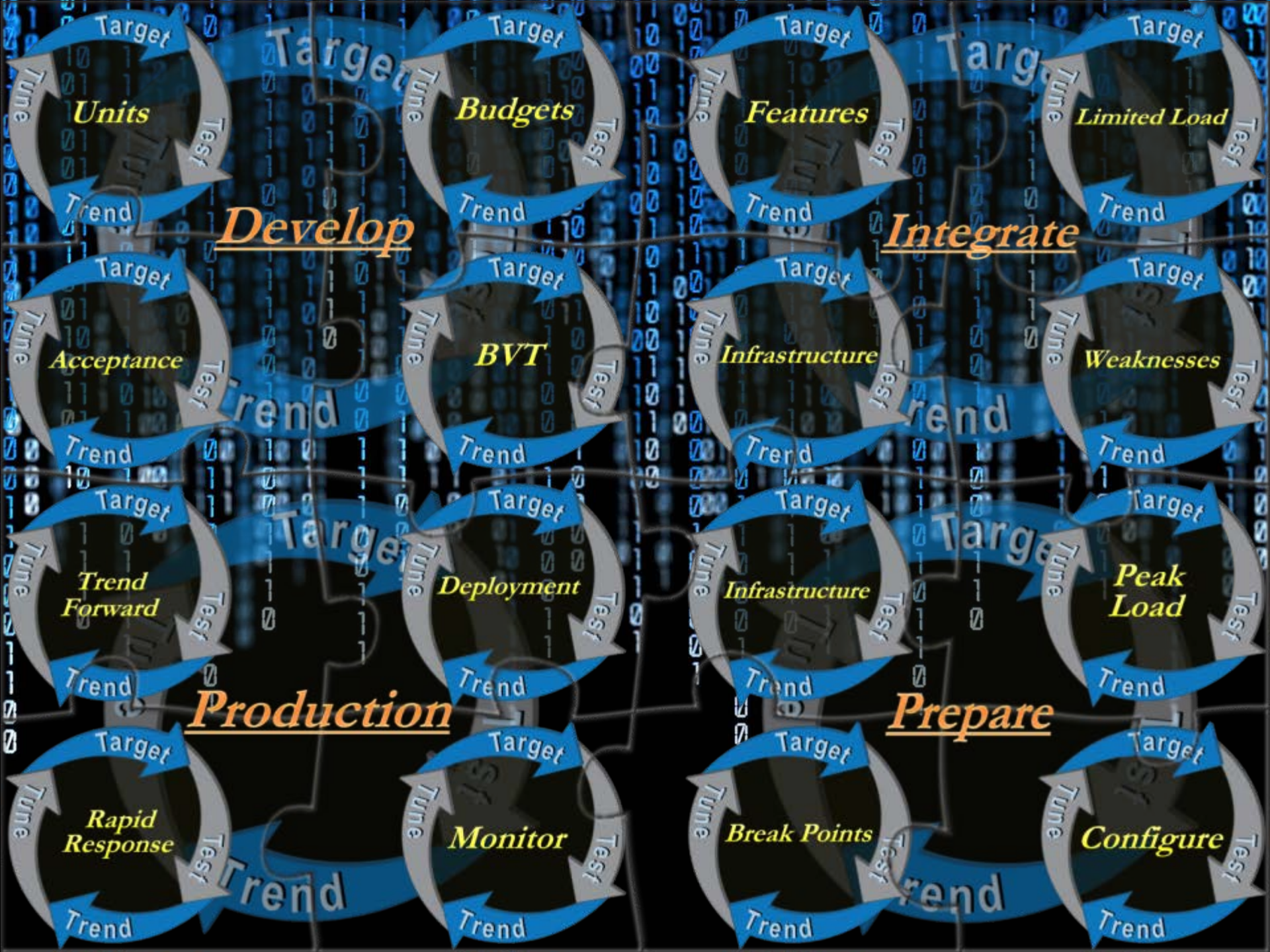
Units
Stories
Tiers
Resources
Goals

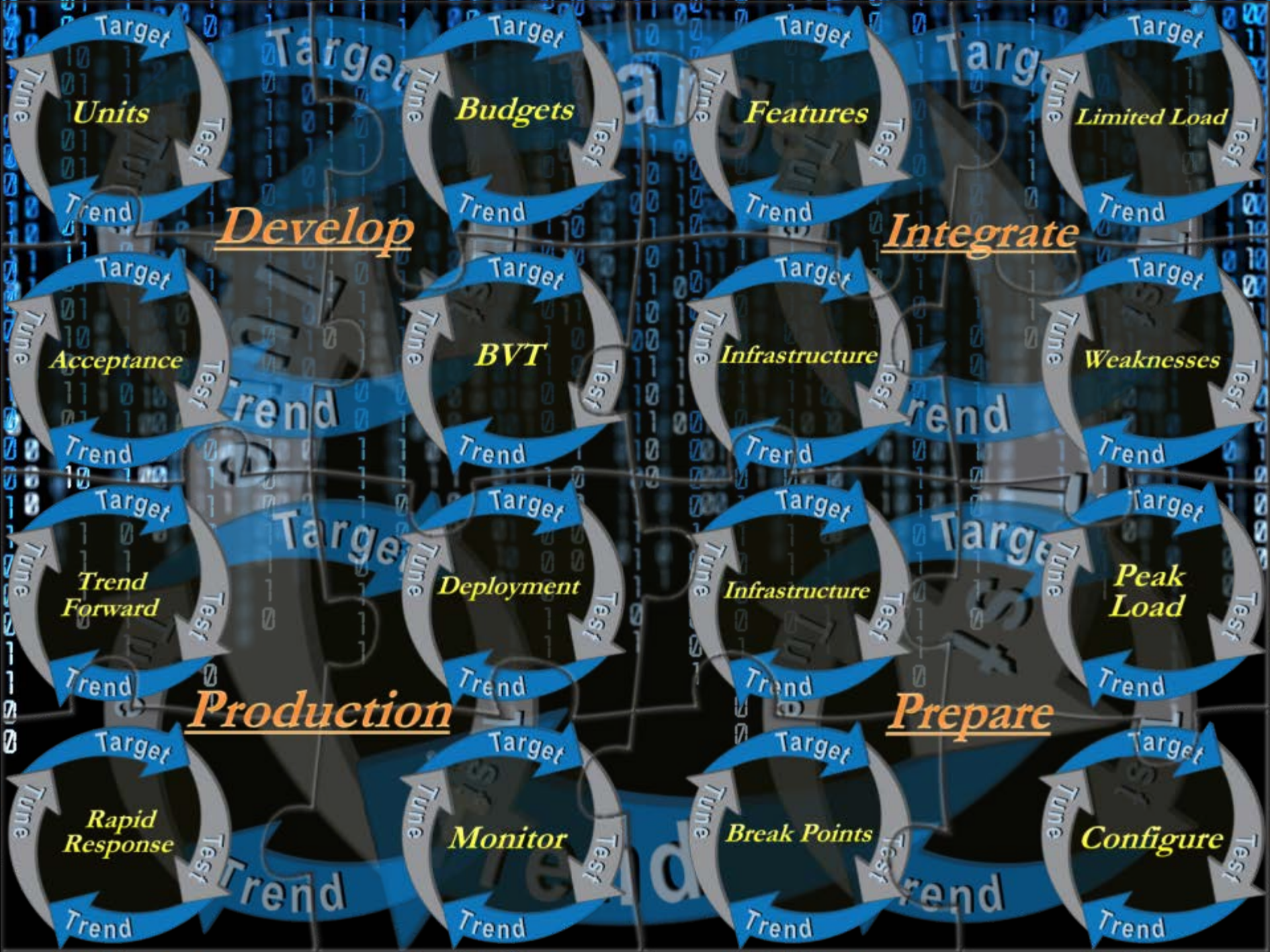
Dev & Prod
Times
Resources
Sizes
Frequencies
Dashboard!

Assess
Compare
Investigate
Accept
Answer


Target
Test
Trend
Tune
Develop
Target
Test
Trend
Tune
Integrate
Target
Test
Trend
Tune
Production
Target
Test
Trend
Tune
Prepare


Develop
Units
Budgets
Features
Limited Load
Integrate
Acceptance
BVT
Infrastructure
Weaknesses
Trend Forward
Deployment
Infrastructure
Peak Load
Production
Prepare
Rapid Response
Monitor
Break Points
Configure
Target
Test
Trend
Tune


Develop
Units
Budgets
Integrate
Features
Limited Load
Acceptance
BVT
Infrastructure
Weaknesses
Production
Trend Forward
Deployment
Infrastructure
Peak Load
Prepare
Rapid Response
Monitor
Break Points
Configure
Target
Test
Tune
Trend


Develop
Units
Budgets
Integrate
Features
Limited Load
Acceptance
BVT
Infrastructure
Weaknesses
Production
Trend Forward
Deployment
Infrastructure
Peak Load
Prepare
Rapid Response
Monitor
Break Points
Configure
Target
Test
Tune
Trend

Value Proposition

A Few Minutes, Every Day

Minimizes Surprises

Methodology / Tool Agnostic

Adapts to Current Priority or Risk

Big Visible Dashboards

No Purchase Necessary

Collaborative

Simplifies Root-Cause Analysis

Challenges

Some Education Required

Demands Discipline

Culture Change is Hard

Several Cycles to Full Value

An Ounce of Prevention…

"What fits your busy schedule better, exercising one hour a day or being dead 24 hours a day?"

APM™

Preventing Poor Performance with
a *little* work…
***every* day…**
from *every* one.

Unit-Level Testing Tools

(‘cause folks always ask)

FireBenchmarks; Performance testing addin for NUnit

JUnitPerf; a collection of JUnit test decorators for performance

Firefox Performance Tester's Pack

HTTPerf

Questions b4 Coffee?

While you're getting coffee…

Discuss with the people near you:

Not counting load-related questions…

What questions have you been asked recently about performance?

How often was finding the answer more effort than seemed valuable?

Can you think of any ways to find answers more quickly/easily?

Remember, if you wish to make notes:

http://typewith.me/p/NEXT2012

I will ask a few volunteers to share their thoughts

Random ?s are often the result of Hearsay

"No, go ahead. I enjoy good gossipy hearsay."

How do *I* address *random* performance questions?

I use…

Rapid Performance Testing

Essential Information for the Entire Team

Evolved from:

*"What have we **got?***
*What do we **want?***
***How** do we get there…?"*

--Bob Barber (Scott's dad)

*… as **quickly**, **simply**, and **cheaply** as possible?*

--Addendum added by: *Scott Barber*

Which is…

…a common man's way of expressing the problem solving approach that classical engineers employ.

- **Given:** *"What have we got?"*
- **Find:** *"What do we want?"*
- **Solve:** *"How do we get there?"*

Premise

Value Begins with Clear Objectives

*What **value** do we hope to gain?*

RPT questions are often not known requirements, goals, thresholds, or constraints

*Value should be the **main driver** behind performance test design and planning*

RPT questions often indicate the true priorities of stakeholders

*RPT answers will frequently **override requirements** in "go-live" decisions*

Attribution

RPT is:

*Inspired by **Rapid Software Testing***

Consistent with Rapid Software Testing themes

Sanctioned** by James Bach, Michael Bolton & the RST instructors to **as a specific implementation of the Rapid Testing Methodology

For more information about RST, visit:
http://www.satisfice.com/info_rst.shtml

What is it?

An approach to respond to a specific performance-related question after 4 or fewer hours of team effort with 1 or more of:

A) The answer

B) A partial answer

- *To determine the value of additional effort*
- *The level of effort to provide the answer*

C) Better questions to address the underlying concern

Conceptual Approach

1. *Receive Question*
   - *Clarify the question*
   - *Understand the driver(s) behind the question*
2. *Generate Test Coverage Outline (TCO)* *(~20 min)*
   - *Simplest path to (partial) answer(s)*
   - *Comprehensive path to (partial) answer(s)*
3. *Transform TCO into Rapid Strategy* *(~20 min)*
   - *Only tasks that fit in time box*
   - *Stick to tasks requiring available resources*

Conceptual Approach

1. Execute Strategy (~2.5 hrs)
   - Snapshots are your friends
   - Anecdotal is sufficient
2. Consolidate/Analyze Data (~30 min)
   - Identify patterns
   - Confirm patterns (time permitting)
3. Report Results (~20 min)
   - Answer(s)
   - Time/Effort to answer(s)
   - Follow-on questions of interest

Questions b4 Tools & Examples?

Tools & Examples

Click Me!!!

While you're at lunch…

Discuss with the people near you:

Do you think your team/organization would benefit from implementing T4 APM™ and RPT ?

How would your Load Testing program be effected if your org implemented T4 APM™ and RPT ?

If you were a project manager and had to choose T4 APM™ + RPT ***or*** Load Testing, which would you choose & why?

Remember, if you wish to make notes:

http://typewith.me/p/NEXT2012

I will ask a few volunteers to share their thoughts

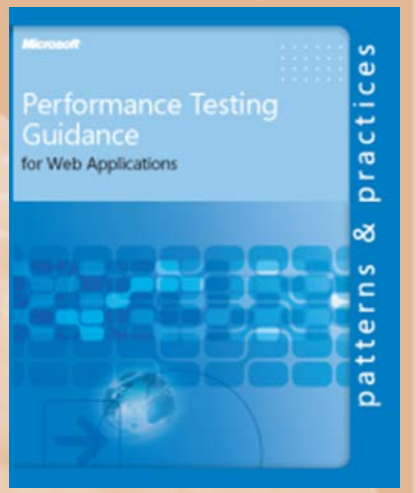
T4 APM™ + PPT *or* Load Testing?

Jessica's Story

Heuristic Load Testing

Essential Information for the Entire Team

Credits



Some of this material was developed for, or inspired by, *Performance Testing Guidance for Web Applications*, a Microsoft patterns & practices book by J.D. Meier, Scott Barber, Carlos Farre, Prashant Bansode, and Dennis Rea.

Many ideas in this course were inspired or enhanced by colleagues including Alberto Savoia, Roland Stens, Richard Leeke, Mike Kelly, Nate White, Rob Sabourin, Chris Loosley, Ross Collard, Jon Bach, James Bach, Jerry Weinberg, Cem Kaner, Dawn Haynes, Karen Johnson, and the entire WOPR community.

Most of the concepts in this presentation are derived from publications, presentations, and research written and/or conducted by Scott Barber.

Many ideas were improved by students who took previous versions of this course, back to 2001.

This course has been heavily influenced by:

Rapid Software Testing (James Bach & Michael Bolton, ©1995-2012 Satisfice, Inc.)

Just-In-Time Testing (Robert Sabourin, ©1998-2012 Amibug, Inc.)

Primary Goal

To teach you how to *think about*, *organize*, and *manage* load testing effectively, under time and resource constraints, by examining nine *core principles* common to successful load testing projects and examining how you can rapidly apply those principles to your project *context*.

Secondary Goal

To introduce you to how to apply *heuristics* and *oracles* to increase your ability to more *efficiently* and *effectively* achieving the objectives of your load testing projects.

Heuristics

heu·ris·tic Pronunciation [hy*oo*-**ris**-tik]

–adjective

1. encouraging a person to learn, discover, understand, or solve problems on his or her own, as by experimenting, evaluating possible answers or solutions, or by trial and error: a heuristic teaching method.

Guideword Heuristics in this presentation…

- **were assembled to help Scott organize and remember stuff.**
- **are fallible (ask Scott about times when they have failed him).**
- **are incomplete.**
- **are not all relevant to every project and every context.**

If these heuristics don't work for you, create your own (and update the mnemonic)!!

Heuristics ≠ Process

A heuristic is not an *edict*. Heuristics require guidance and control of skilled practitioner.

Heuristics are context-dependent.

Heuristics may be useful even when they contradict each other– especially when they do!

Heuristics can substitute for complete and rigorous analysis.

Slide Adapted from *Rapid Software Testing* by James Bach & Michael Bolton, © 1995-2007, Satisfice, Inc.

Load Testing Principles:

CCD IS EARL

(A mnemonic of guideword heuristics)

Load Testing Principles

Context	Project context is central to successful testing.
Criteria	Business, project, system, & user success criteria.
Design	Identify system usage, and key metrics; plan and design tests.
Instrument	Install and prepare environment, tools, & resource monitors.
Script	Script the tests as designed.
Execute	Run and monitor tests. Validate tests, test data, and results.
Analyze	Analyze the data individually and as a cross-functional team.
Report	Consolidate and share results, customized by audience.
Iterate	"Lather, rinse, repeat" as necessary.

Principles ≠ Process

- Context
- Criteria
- Design
- Instrument
- Script
- Execute
- Analyze
- Report
- Iterate

"One-size-fits-all" approaches fit load testing poorly.

Most successful load testing projects involve at least active decision making around these core principles.

Core principles are not sequential, go by many different names, have varying priorities, and may be implicit or explicit.

Core principles are not in themselves an approach or process.

Core principles represent a foundation upon which to build a process or approach based on the context of your project.

Context

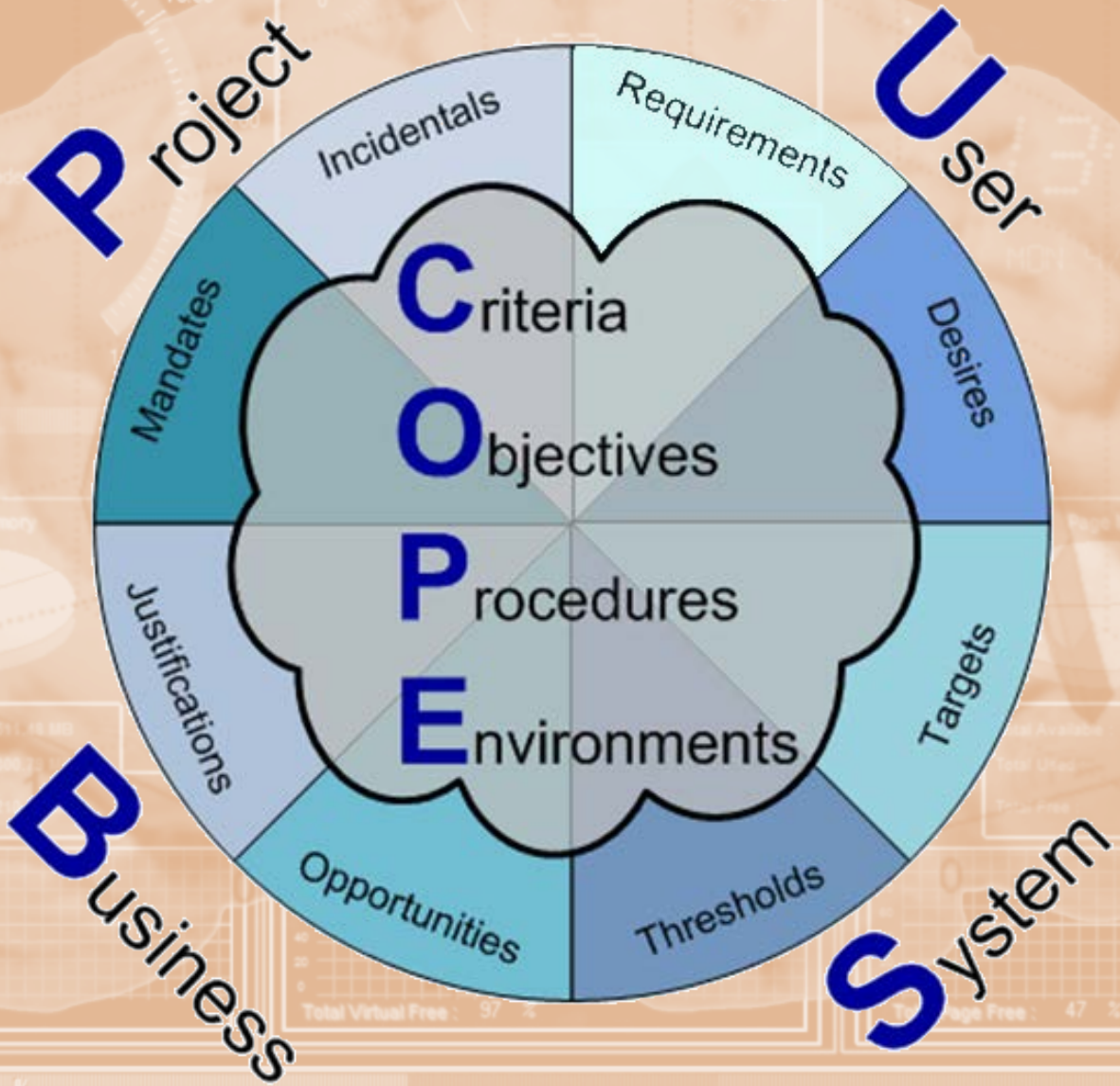
Context

When assessing project context, I

COPE in PUBS

(Another mnemonic of guideword heuristics)

Context



Context

Do you know your performance testing mission?

Do you know the “Commander’s Intent”?

Can you find out?

Might COPE in PUBS help?

Example from my days as a U.S. Army LT:

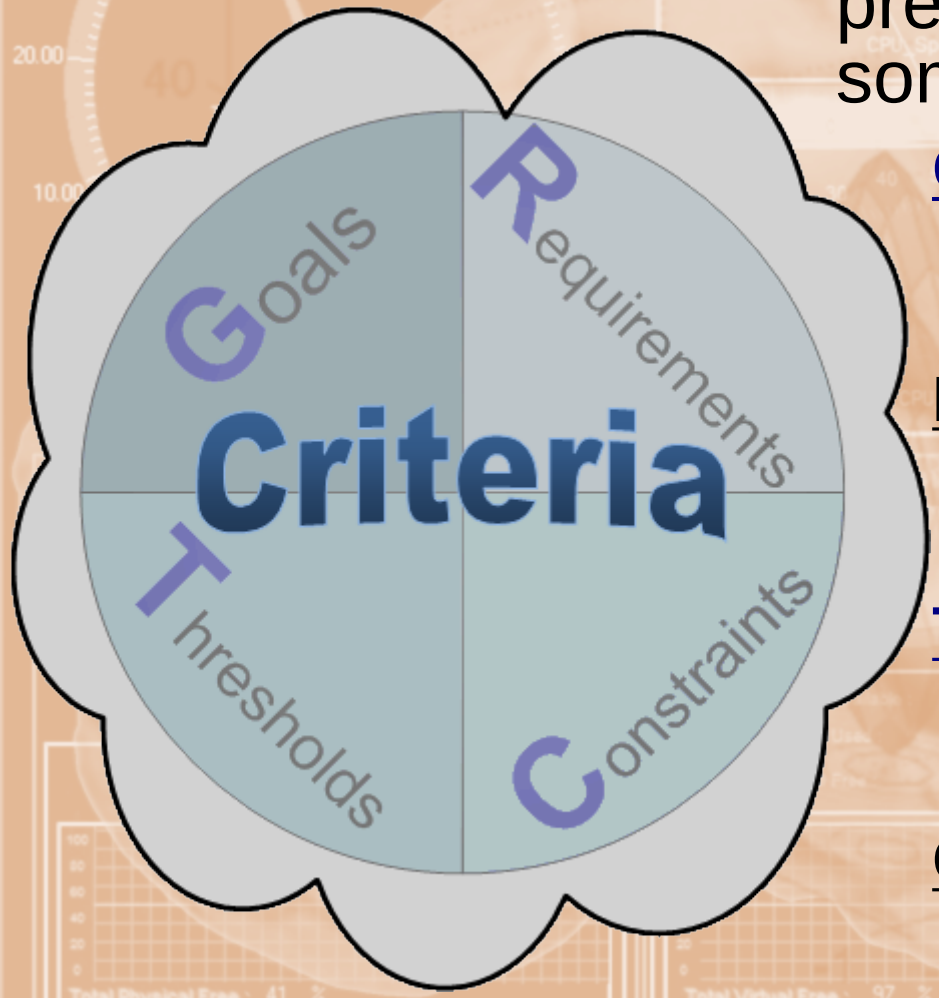
Mission: Secure hilltop 42 NLT 0545 tomorrow.

Commander’s Intent: It is my intent that the supply convoy safely cross the bridge spanning the gorge between hilltop 42 and hilltop 57 between 0553 and 0558 tomorrow.

Criteria

Criteria

Performance Criteria are *boundaries* dictated or presumed by someone or something that matters.



Goals: Soft Boundaries (User Satisfaction)

Requirements: Firm Boundaries (Business or Legal)

Thresholds: Hard Boundaries (Laws of Physics)

Constraints: Arbitrary Boundaries (Budget or Timeline)

Criteria

Context + Criteria =>

Load Testing Objectives

Essential Information for the Entire Team

Load Testing Objectives

What we actually hope to gain by testing performance

Are sometimes completely unrelated to stated requirements, goals, thresholds, or constraints

Should be the main drivers behind performance test design and planning

Usually indicate the performance-related priorities of project stakeholders

Will frequently override goals in “go-live” decisions

*(Hmm… Where have I heard ***that*** before? <grin>)*

Criteria

How do you evaluate criteria?

Know your oracles.

For a video lecture, see:
http://www.satisfice.com/bbst/videos/BBSTORACLES.mp4

Slide Adapted from *Rapid Software Testing* by James Bach & Michael Bolton, © 1995-2007, Satisfice, Inc.

Criteria

An oracle is the principle or mechanism
by which you recognize a problem.

therefore...

Without an oracle you **cannot** recognize a problem

and conversely...

If you think you see a problem,
you **must** be using an oracle.

Slide Adapted from *Rapid Software Testing* by James Bach & Michael Bolton, © 1995-2007, Satisfice, Inc.

Criteria

Consistency ("this agrees with that") *is an important theme in oracles*

History: The present version of the system *is consistent* with past versions of it.

Image: The system *is consistent* with an image that the organization wants to project.

Comparable Products: The system *is consistent* with comparable systems.

Claims: The system *is consistent* with what important people say it's supposed to be.

Users' Expectations: The system *is consistent* with what users want.

Product: System elements *are consistent* with comparable elements in the system.

Purpose: The system *is consistent* with its purposes, both explicit and implicit.

Statutes: The system *is consistent* with applicable laws and legal contracts.

Familiarity: The system ***is not consistent*** with the pattern of any familiar problem.

Consistency heuristics rely on the quality of your models of the product and its context.

Slide Adapted from *Rapid Software Testing* by James Bach & Michael Bolton, © 1995-2007, Satisfice, Inc.

Design

"Enterprise grade load generation tools are designed to look easy in sales demos.

Don't be fooled."

--Scott Barber

Design

To help me decide what tests to design, I use:

VECTRAS

(An acronym of guideword heuristics)

Design

Do I need this test to:

Investigate *or* **Validate/Verify**

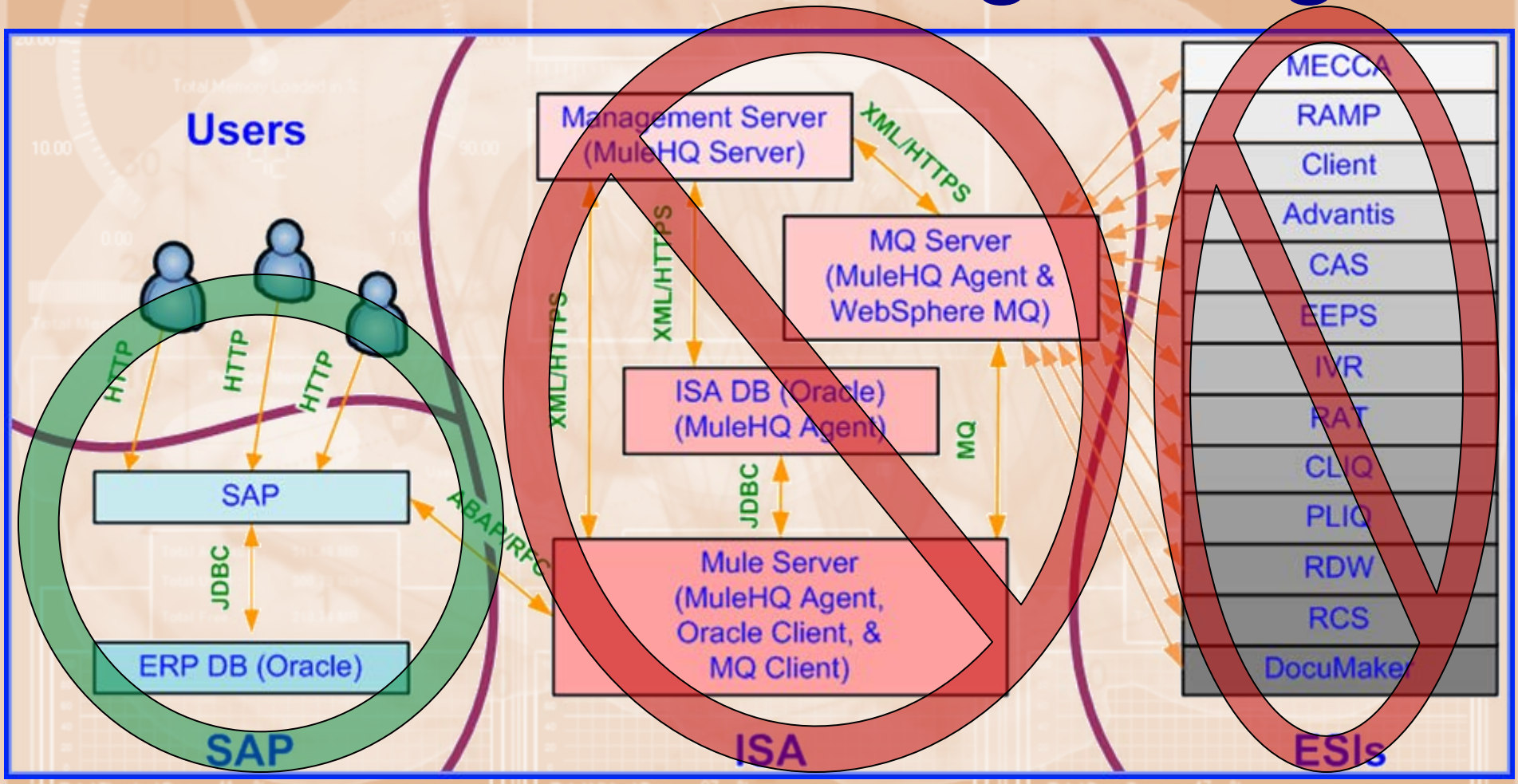
End-to-End *or* **Component**

response **Times** *and/or* **Resources** *utilized*

under **Anticipated** *or* **Stressful** *conditions*

Design

Communicating Design



Design

When Building Usage Models, I

FIB LOTS

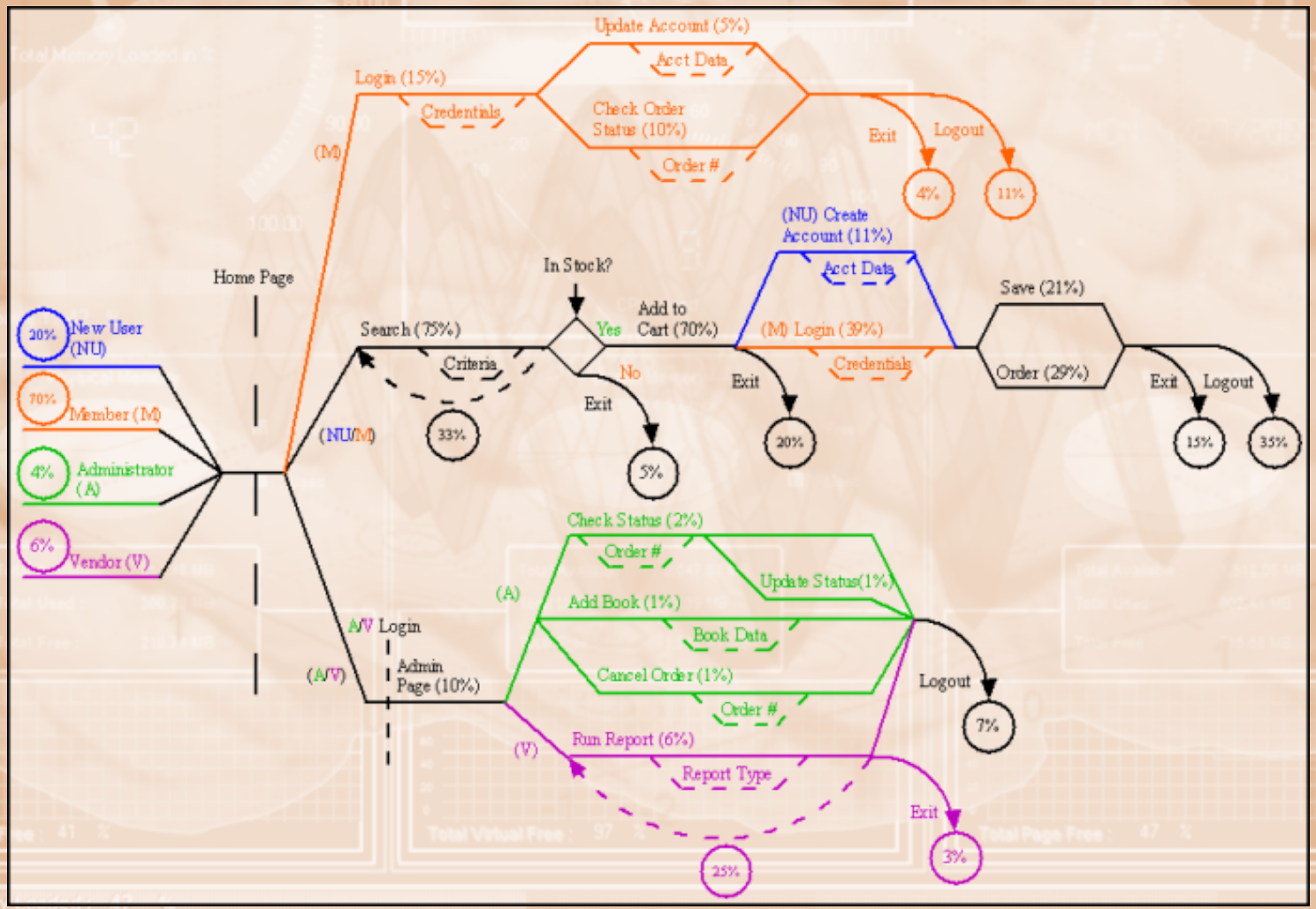
(Yet another mnemonic of guideword heuristics)

Design

Frequent	Common activities (get from logs)
Intensive	e.g. Resource hogs (get from developers/admins)
Business Critical	Even if these activities are both rare and not risky
Legal or Contract	SLA's, Contracts and other stuff that will get you sued
Obvious	What the users will see and are mostly likely to complain about. What is likely to earn you bad press
Technically Risky	New technologies, old technologies, places where it's failed before, previously under-tested areas
Stakeholder Mandate	Don't argue with the boss (too much)

Design

Communicating System Usage



Design

I Approach Front-End Testing With

SCORN

(Another mnemonic of guideword heuristics – you hope!)

Design

Size

Media, HTML, styles & scripts – compress & minify.

Caching

The end-user's browser cache can be your best friend, or your worst nightmare, use it wisely.

Order

Get the load order of your scripts and styles wrong, and you'll lose your users every time – even though response time hasn't changed!

Response Codes

3, 4, & 5xx series response codes on individual objects are bad things.

Number

When it comes to performance, less is more (usually).

Design

Apply Experimental Theory/Design

Start with a *Hypothesis*

- Design a test to try to show how it *might* be true under *some* condition
- These are good RPT candidates.

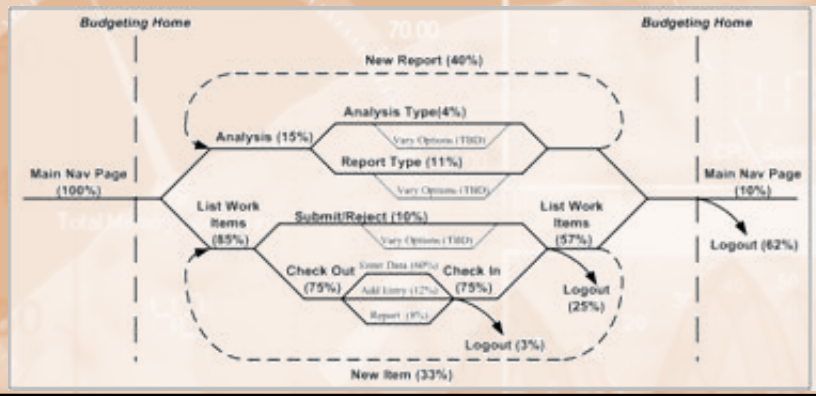
If true under some condition, it's now a *Theory*

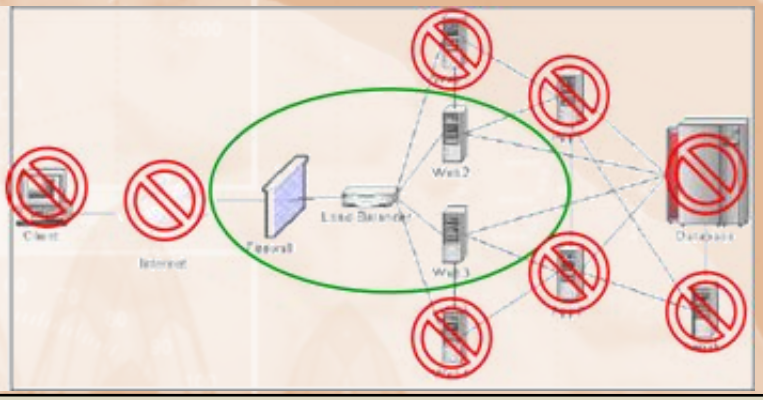
- Design tests to try to show conditions under which it is *not* true
- Where most of Load Testing time is spent

If you cannot disprove it, think of it as a *Law*

- Treat it as true until a condition changes or you get more information… then design new tests.
- New info often results from T4 APM™

Design

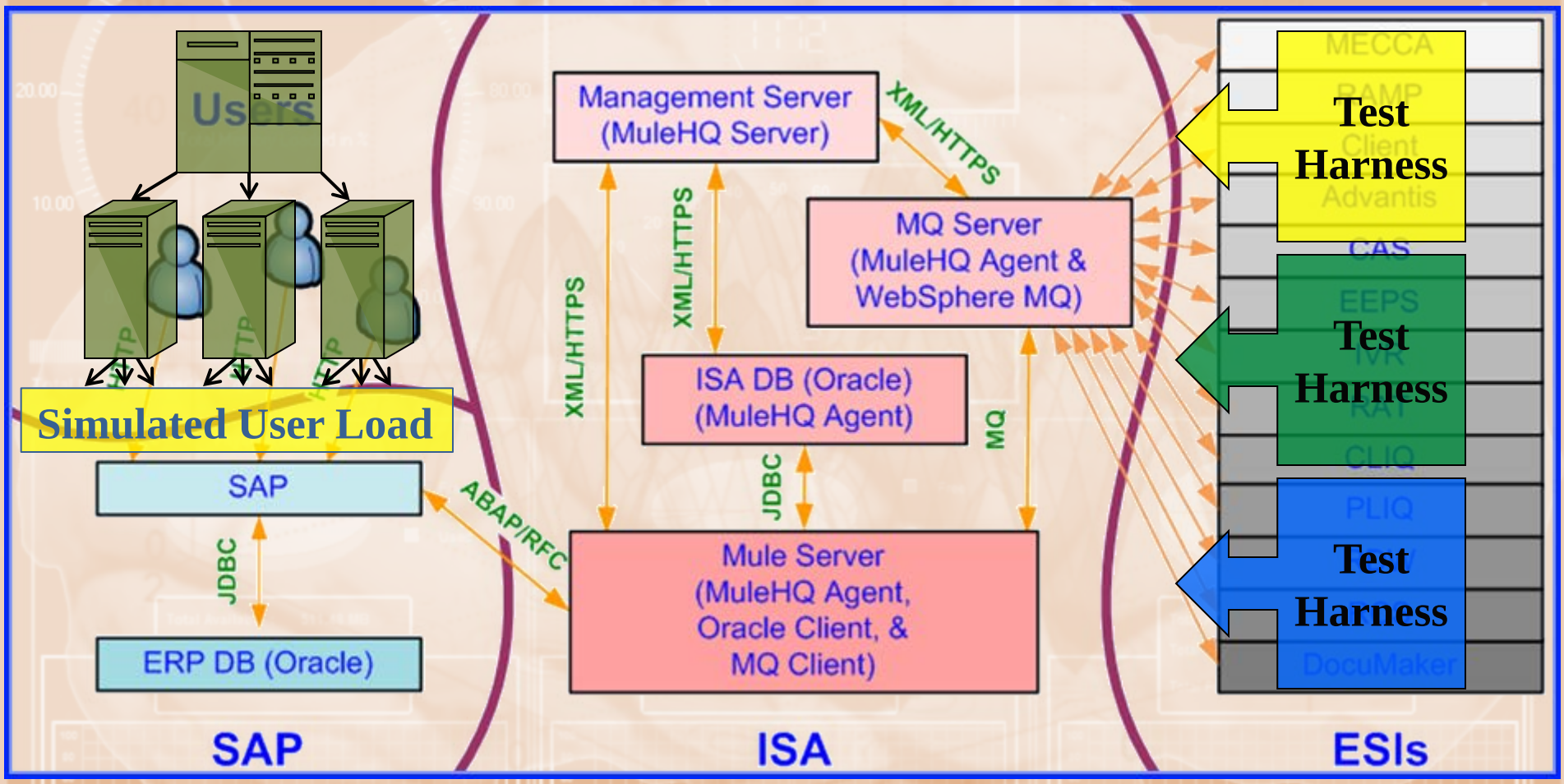




Intent of Investigation:	Collect configuration data for tuning. Collect data to assist in validating existing network.
Prerequisites:	Static prototype deployed on future production hardware.
Tasks:	Determine network bandwidth, validate firewalls & load balancer, evaluate web server settings.
Tools & Scripts:	Load generation tool, HTTP scripts to request objects of various sizes from a pool of IP addresses.
External Resources Needed:	Firewall, Load Balancer, Network Admins, network monitors, 20 IP addresses for spoofing.
Risks:	Schedule delay, availability of administrators, configuration of load generation tool for IP spoofing,
Data of Special Interest:	Network bandwidth & latency, load balancer effectiveness, resource consumption, response times.
Areas of Concern:	No internal expertise on load balancer configuration.
Pass/Fail Criteria:	Adequate available bandwidth, architectural assumptions validated.
Completion Criteria:	Critical data collected and assumptions validated.
Planned Variants:	1 to 20 IPs, volume of 1 to 500, size from 1Kb to 1mb, configuration settings.
Execution Duration(s):	6 days: 2 days ea. network & bandwidth, firewall and load balancer, web server configuration.

Instrument

Instrument



Instrument

Install	SUT, Load Generator, Monitors, Tools, Utilities, Processes, & Team
Configure	Installations, Data, Hooks, Stubs, Harnesses, & Schedule
Validate	Configurations, Assumptions, Design, Models, Integrations, Gaps, & Support
Adapt & Adjust	Everything so far based on Validations.
Coordinate	Prepare for execution

Script

"MacGyver is a super-hero,

****not****

a career path."

--Scott Barber

Script

When creating scripts, I try to:

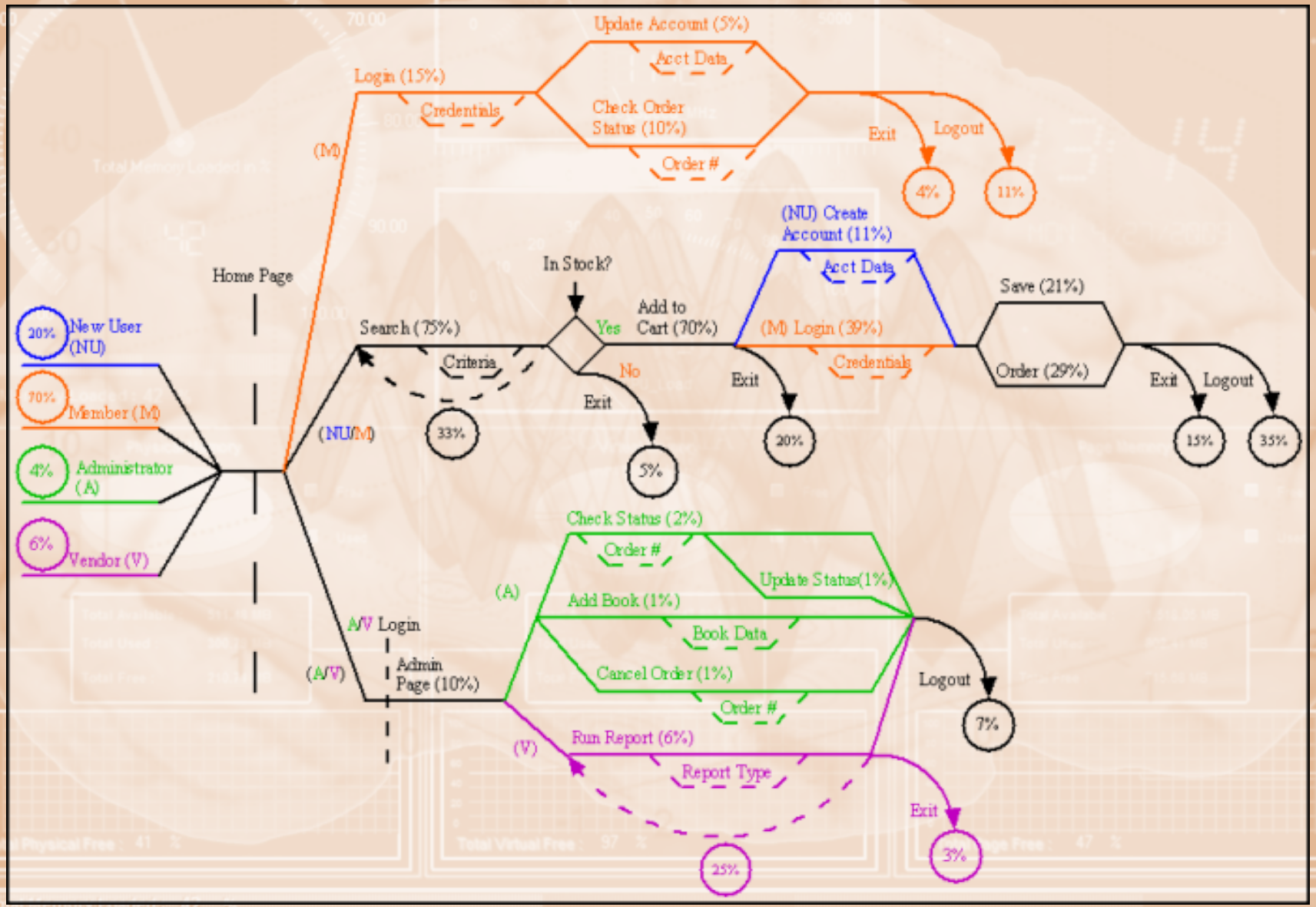
FIND HARM

(Yet another mnemonic of guideword heuristics)

Script

Functionality	Ensure obvious functional errors are detected
Input	Consider “pre-script” validation and/or transformation
Navigation	If you think there are 3 ways to do something, users will find 5… the other two could be performance killers
Data	Vary data to avoid unintentional caching and determine the difference between “speed” and “volume” effects
Human variability	“Super-users” yield “Stinky-scripts”
Abandon	Not logging out can leave resource consuming session artifacts
Ramping & Marching	The step model, is not just unrealistic, it can invalidate results
Maintainability	If you don’t design your scripts to be maintainable, they won’t be maintained

Script



Script

Load generation tools

- Do not interact with client side portions of the application.
- Do not natively evaluate correctness of returned pages.
- Often don't handle conditional navigation.
- Do not handle abandonment well.

Scripting concepts

- Record – EDIT – playback
- Add data variance
- Add delays
- Add conditional logic
- Add code to evaluate correctness of key pages
- Add abandonment functions

Script

Real Users React

Ensure your tests represent the fact that real users react to the application.

Vary Data

Make sure that data being entered is unique for each simulated user.

Make sure that each simulated users is unique (this may mean more than just separate IDs and Passwords).

Vary Navigation Paths

If there is more than one way for a user to accomplish a task in the application, your test must represent that.

Different paths through the system often stress different parts of the system.

Script

Users Think… and Type

- Guess what? They all do it at different speeds!
- Guess what else? It's your job to figure out how to model and script those varying speeds.

Determine how long they think

- Log files
- Industry research
- Observation
- Educated guess/Intuition
- Combinations are best

Script

Abandonment

If a page takes too long to display, users will eventually abandon your site – thus lessening the load – changing the overall performance.

Not simulating abandonment makes your test un-intentionally more stressful than real life.

Page Name	Abandonment Distribution	Abandonment Min Time	Absolute Abandonment
Home Page	Normal	5 sec	30 sec
Pay Bill	Uniform	10 sec	240 sec
Search Web	Negexp	8 sec	30 sec
Submit Taxes	Inverse Negexp	30 sec	900 sec
Validate Field	Normal	5.5 sec	20 sec

Script

Delays

Every page has a think time – after you determine the think time for that page, document it.

These think times should cause your script to pace like real users.

Event Type	Event Name	Type	Min	Max	Std	Req't	Goal
Procedure name:	Initial Navigation()						
Timer name:	tmr_home_page	negex p	4	N/A	N/A	8	5
Timer name:	tmr_login	normdist	2	18	4.5	8	5
Timer name:	tmr_page1	linear	5	35	N/A	8	5
Timer name:	tmr_data_entry	negex p	8	N/A	N/A	8	5
Timer name:	tmr_page2	normdist	3	9	3	5	3
Timer name:	tmr_submit_transaction	linear	2	4	N/A	5	3
Timer name:	tmr_signout	N/A	N/A	N/A	N/A	8	5

Execute

“There is no such thing as a ‘junior performance tester’…

but there are people who are new to performance testing.“

--Scott Barber

Execute

To remind me that execution doesn't simply mean "break it", I recall:

ET TO BRUTE

(A mnemonic of guideword heuristics)

(Not to mention an oddly placed Shakespeare reference.)

Execute

Evaluate	Determine what the script(s) actually do (Accuracy).
Test	Check script(s), data, etc. for consistency (Precision).
Trend	If it's worth checking more than once, it's likely worth trending.
Oscillate	Vary between alternate extremes over a definable period.
Baseline	Establishing an understood, reliable point of reference.
Ramp	Increase the load systematically until learning stops.
Unanticipated	Usage won't be exactly what you think, ask "what if…?"
Tune	Work as a collaborative, cross-functional team.
Exploit	When something looks odd "beat on it to see if it breaks".

Execute

Some of Scott's Execution Heuristics:

- 1, 3, 7, 11, More
- Best, Expected, Worst
- Tacoma Narrows
- Justin Beiber's Haircut
- Nancy Knitting
- UAT under load
- If I can't break it, I don't understand it

Analyze

"With an order of magnitude fewer variables performance testing could be a science, but for now,

performance testing is at best a scientific art."

--Scott Barber

Analyze

When I'm analyzing, I remind myself to:

C' STOP & CARE

(A mnemonic of guideword heuristics)

Analyze

Configurations	Results are meaningless without technical context.
Significance & Repeatability	Don’t over-trust results until you can repeat them.
Trends	Within the test run, across tests, across data, etc.
Outliers	If you can repeat it or it’s >1%, it’s not an outlier.
Patterns	Graph, blink, overlay, compare, and contrast.
Compliance	If it can get you sued, check it every time.
Accuracy	How well do the results represent reality.
Resources & Times	This is where users care and symptoms are found.
Errors & Functionality	If it’s broken, performance doesn’t matter.

Analyze

Some Methods

- Blink
- De-Focus & Re-Focus
- Overlay
- Plot
- Bucket
- Look for Odd
- Be Derivative
- Ditch the Digits
- Un-average Averages
- Manual

Analyze

Facts

- Analysis is a team sport.
- We cannot ***prove*** anything.
- Focus on patterns, trends, and feelings.
- Numbers are meaningless out of context.
- Qualitative feedback is ***at least*** as relevant as quantitative feedback.

Analyze

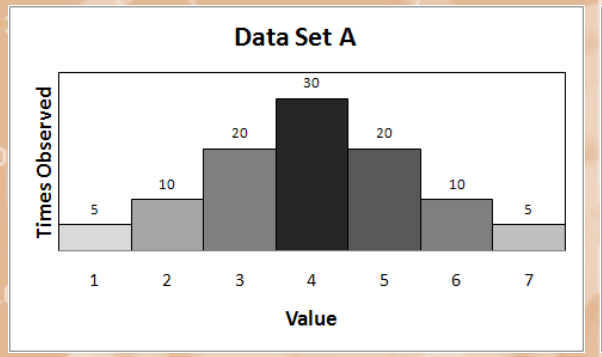
	Sample Size	Minimum	Maximum	Average	Median	Normal	Mode	95th Percentile	Standard Deviation
Data Set A	100	1	7	4	4	4	4	6	1.5
Data Set B	100	1	16	4	1	3	1	16	6.0
Data Set C	100	0	8	4	4	1	3	8	2.6

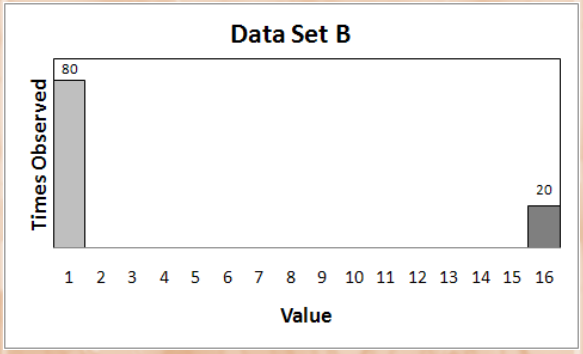
All three have an average of 4.

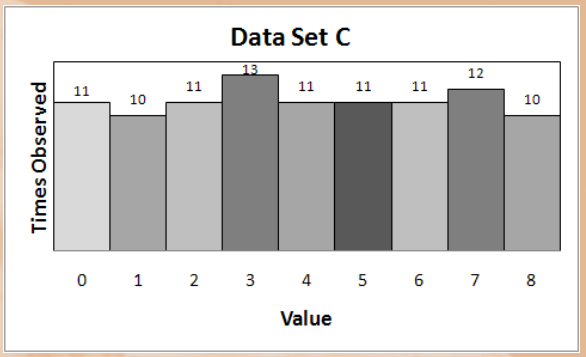
Which has the “best” performance”?

How do you know?

Analyze







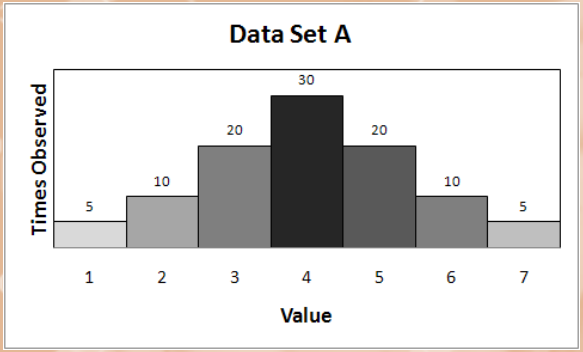
	Sample Size	Minimum	Maximum	Average	Median	Normal	Mode	95th Percentile	Standard Deviation
Data Set A	100	1	7	4	4	4	4	6	1.5
Data Set B	100	1	16	4	1	3	1	16	6.0
Data Set C	100	0	8	4	4	1	3	8	2.6

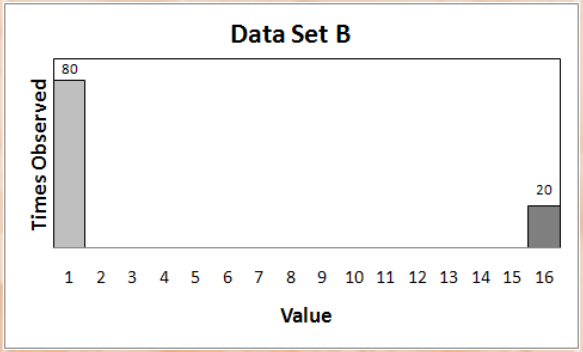
All three have an average of 4.

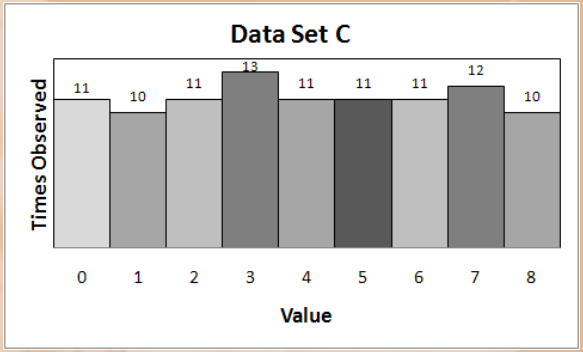
Which has the “best” performance”?

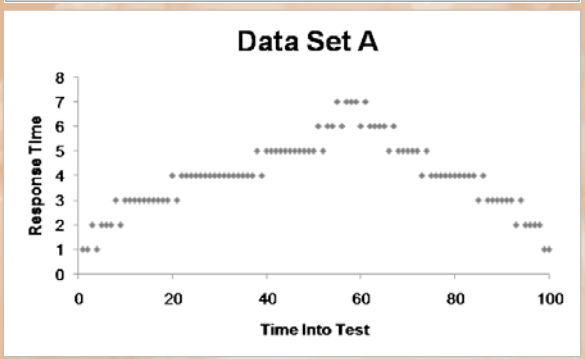
How do you know?

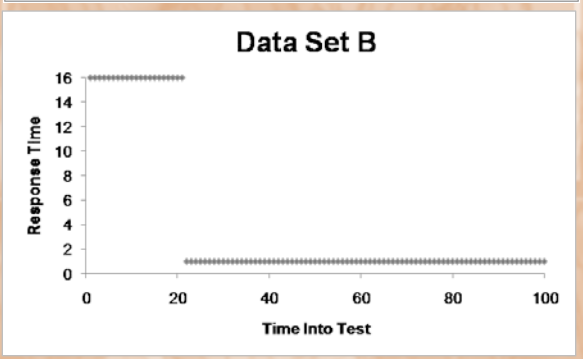
Analyze

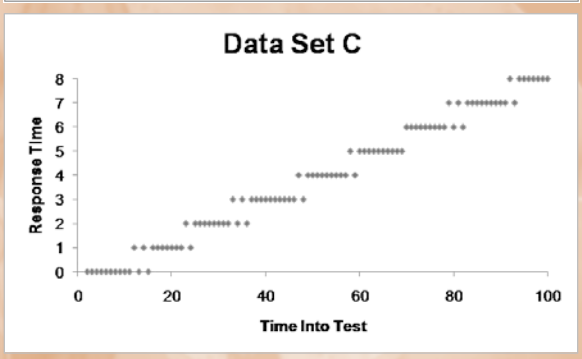












	Sample Size	Minimum	Maximum	Average	Median	Normal	Mode	95th Percentile	Standard Deviation
Data Set A	100	1	7	4	4	4	4	6	1.5
Data Set B	100	1	16	4	1	3	1	16	6.0
Data Set C	100	0	8	4	4	1	3	8	2.6

Now which has the “best” performance”?

Report

Essential Information for the Entire Team

"Linear extrapolation of performance test results is, at best, black magic.

Don't do it (unless your name is Connie Smith, PhD. or Daniel Menasce, PhD.)"

--*Scott Barber*

Report

For good reports, I consult:

TRAVIS

(A mnemonic of guideword heuristics)

Report

Timely	Stakeholders need data to make decisions. Many decisions can't wait until tomorrow.
Relevant	Reports are only interesting if they contain data that is useful.
Audience Appropriate	A great report for developers is probably a lousy report for executives.
Visual	Try to use pictures over numbers and numbers over words. Save words for recommendations.
Intuitive	Strive to make reports compelling without explanation.
Supported	Unless you are hiding something, make the supporting data available to the team.

Report

Facts

- Most people will never read performance test results docs.
- Most people don't really understand the underlying components to performance.
- It is our job to make it easy for them to understand, and understand quickly.
- Being skilled at graphical presentation of technical information is critical for us to help others understand the message we are delivering.
- Confusing charts and tables lead to wrong decisions causing lost $ and ruined reputations.

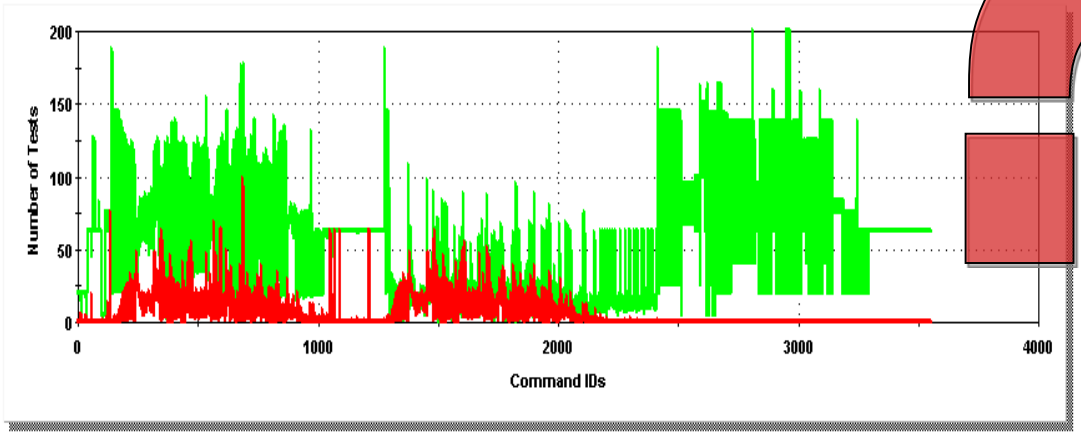
What consumers of reports want

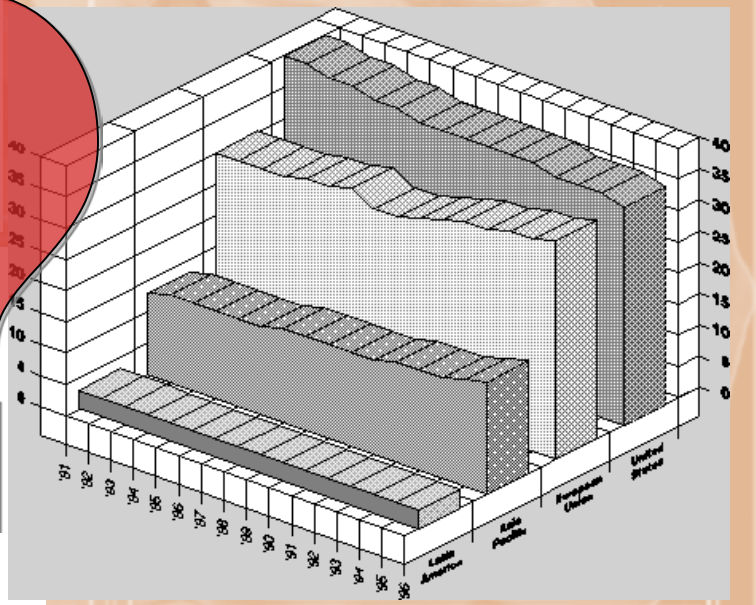
- Answers… NOW! (They might not even know the question)
- To understand information intuitively.
- Simple explanations of technical information.
- To be able to make decisions quickly and have the information to support those decisions.
- “Trigger phrases” to use with others.
- Concise summaries and conclusions.
- Recommendations and options.

Report

What consumers of reports usually get

	Baseline		250		500		750	
Timer Name	95th	Max	95th	Max	95th	Max	95th	Max
ec_Main_Page	10.46	18.09	6.41	8.22	6.33	8.33	3.85	10.84
ec_logon_help	0.98	0.98	0.56	0.59	0.55	0.55	0.49	0.52
ec_login	5.35	7.92	6.66	11.84	6.75	17.03	5.95	14.25
quick_learns	6.66	6.67	5.91	10.98	5.92	11.02	3.92	6.05
view_quick_learn	15.66	17.11	5.53	10.72	3.89	10.61	3.46	10.46
view_faq_window	2.45	2.45	1.47	1.52	1.53	1.66	1.30	6.27
view_faq	0.67	0.67	0.60	0.63	0.58	0.69	0.56	5.41
view_ec_status	8.08	12.55	1.73	6.66	1.80	1.86	1.60	8.34





Report

Strive for something better

- Concise verbal descriptions.
- Well formed, informative charts (pretty pictures).
- Focus on requirements and business issues.
- Don’t be afraid to make recommendations or draw conclusions!
- Make all supporting data available to everyone, all the time (Don’t sit on data ‘cause they won’t understand it).
- Report ≠ Document
- Report *AT LEAST* every 48 hours during execution.

Report

Inspired by “ET”

Edward Tufte, Ph.D., Professor Emeritus of political science, computer science and statistics, and graphic design at Yale.

According to ET:

Power Corrupts...

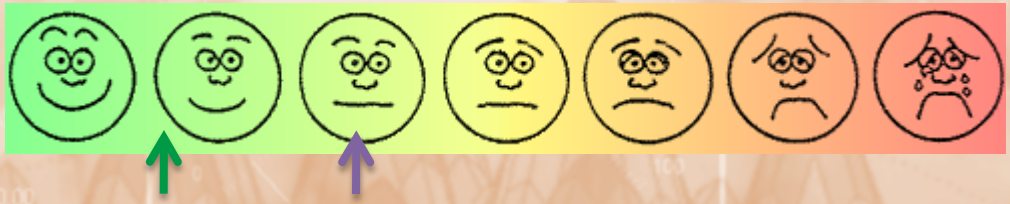
Report

PowerPoint Corrupts Absolutely.

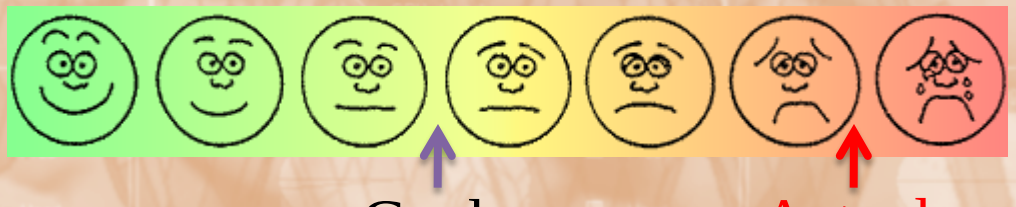
Report

Executive or Business Level

Most pages:



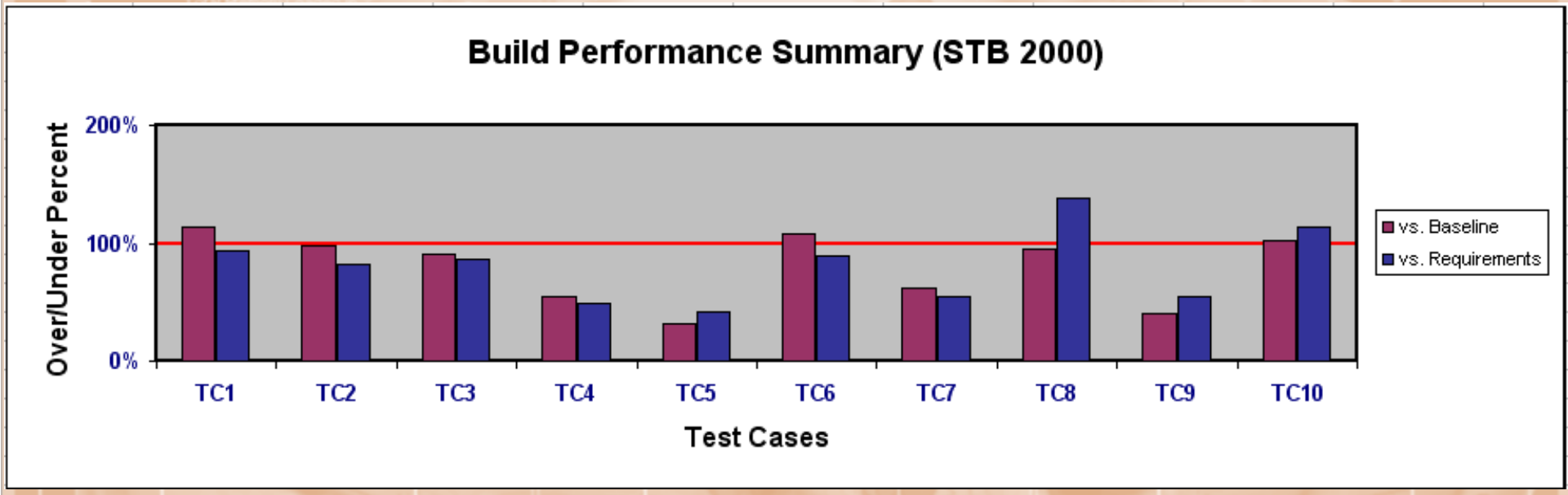
Searches:



"Overall, users are pleased with the performance of everything except searches, which they rated as quite painful."

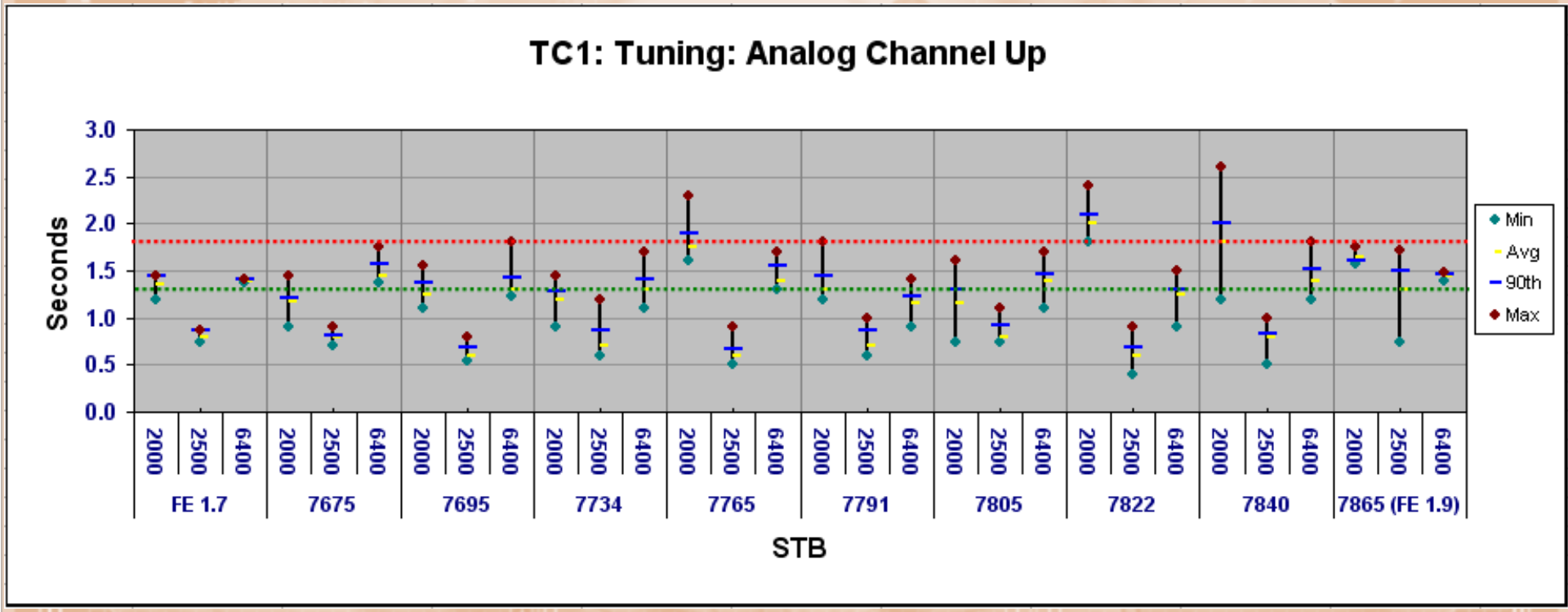
Report

Relative Performance



Report

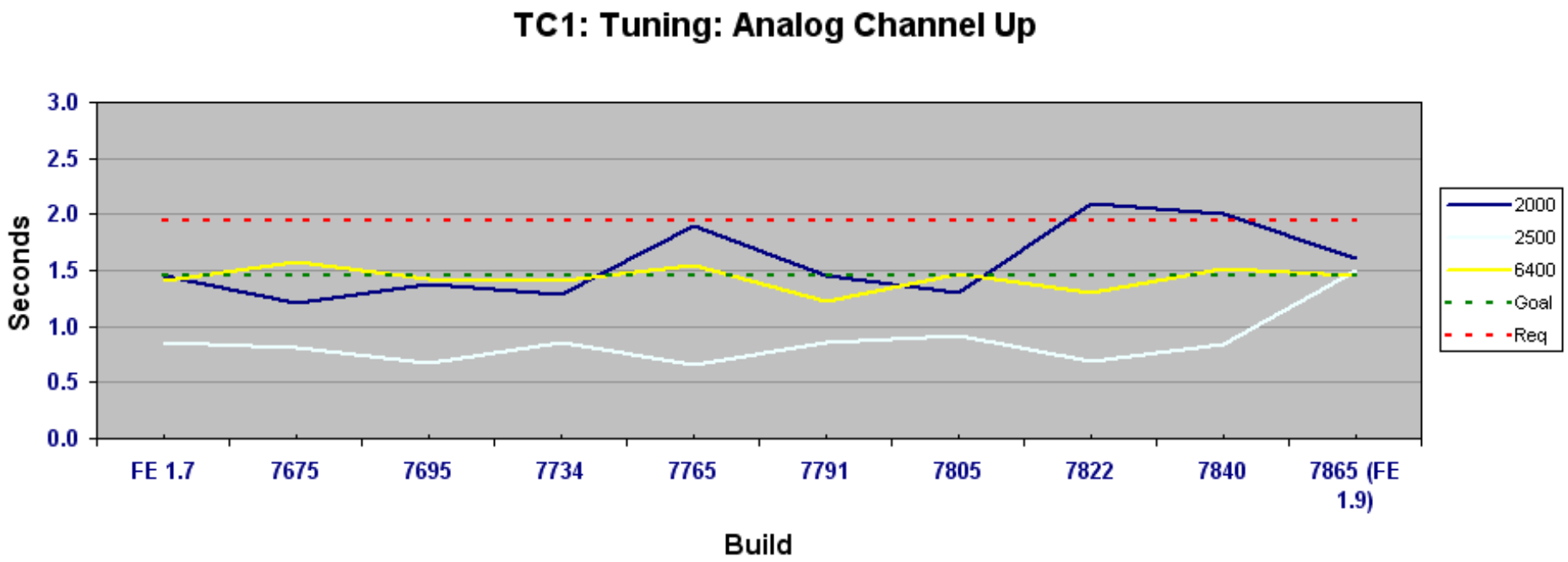
Graphs Make Some Things Obvious…



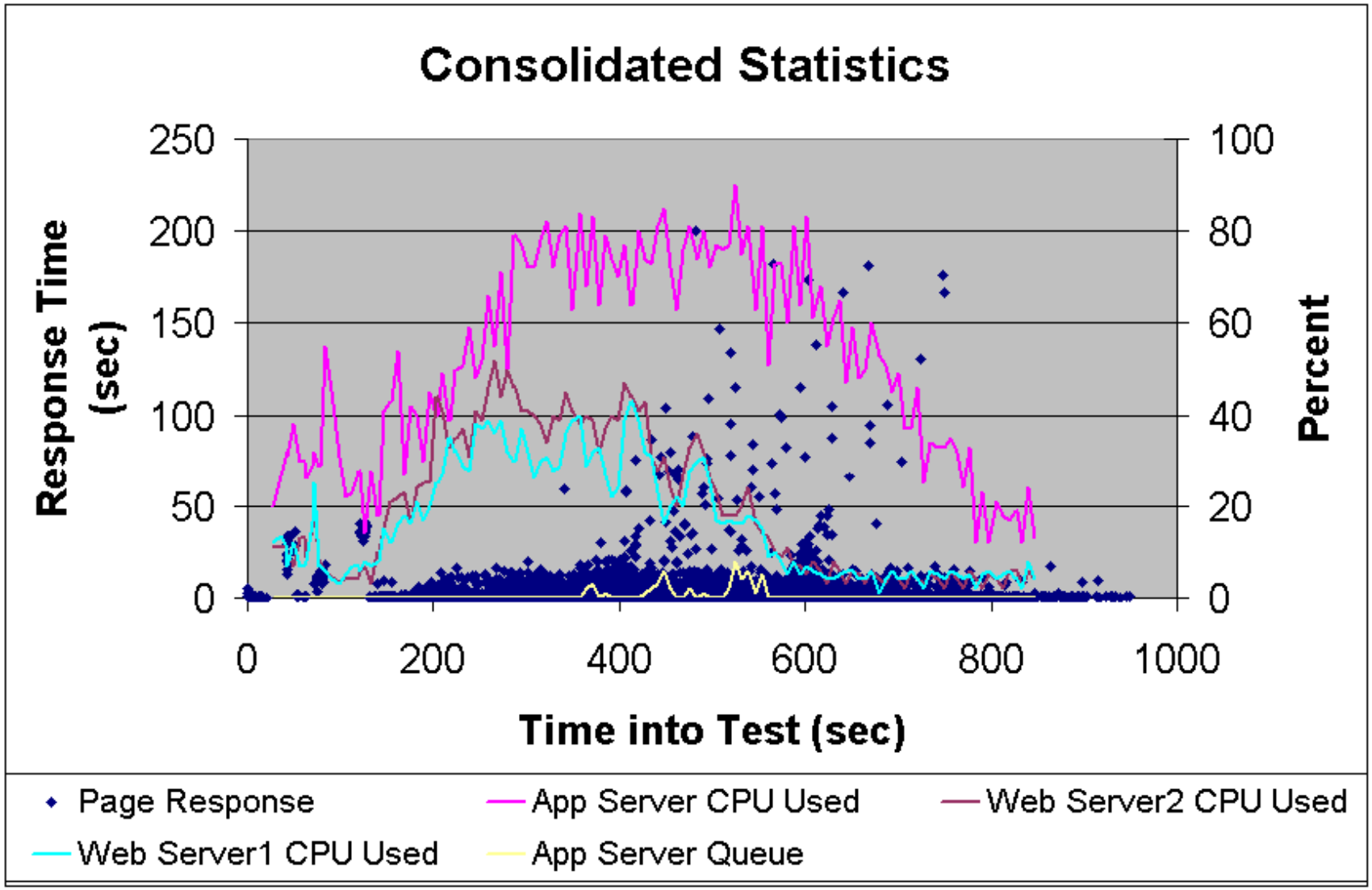
… To Some People

Report

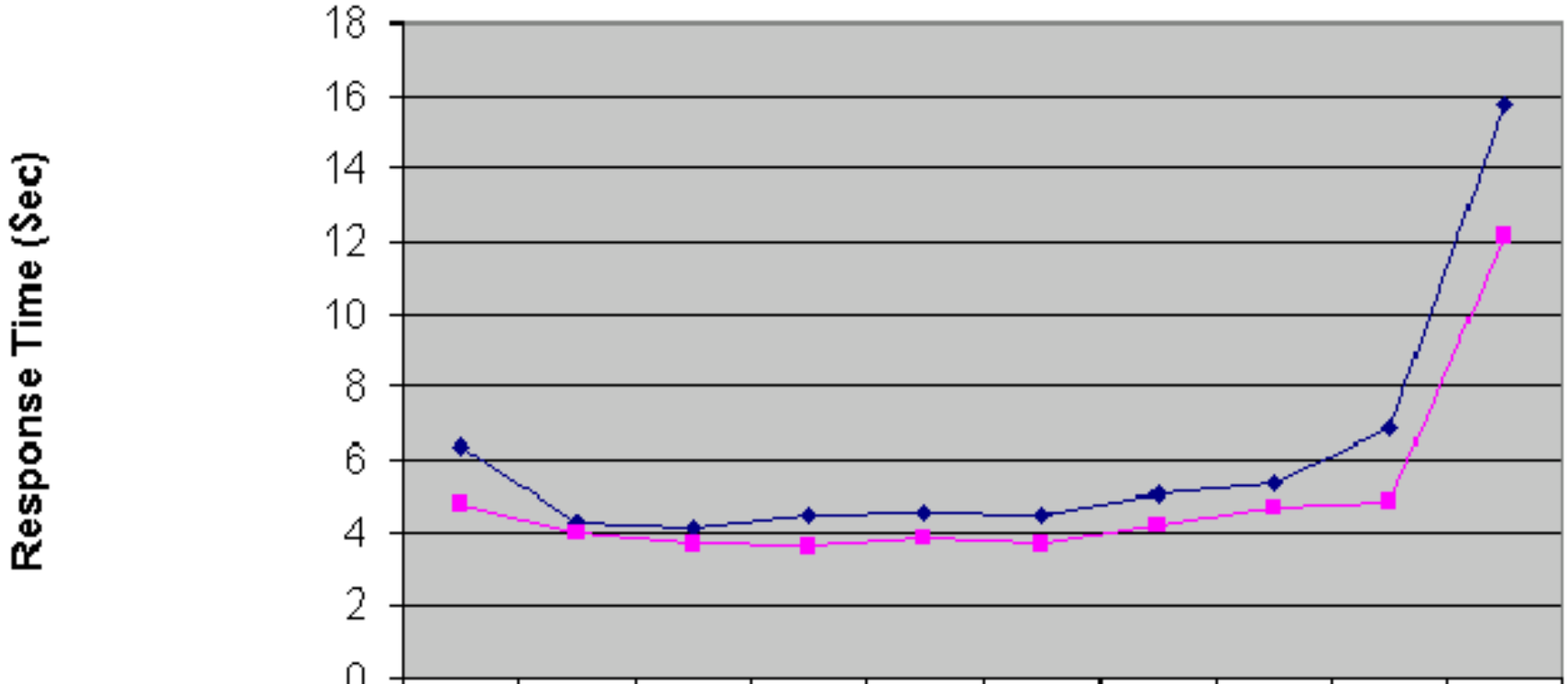
Trends, Trends, Trends!!!



Report

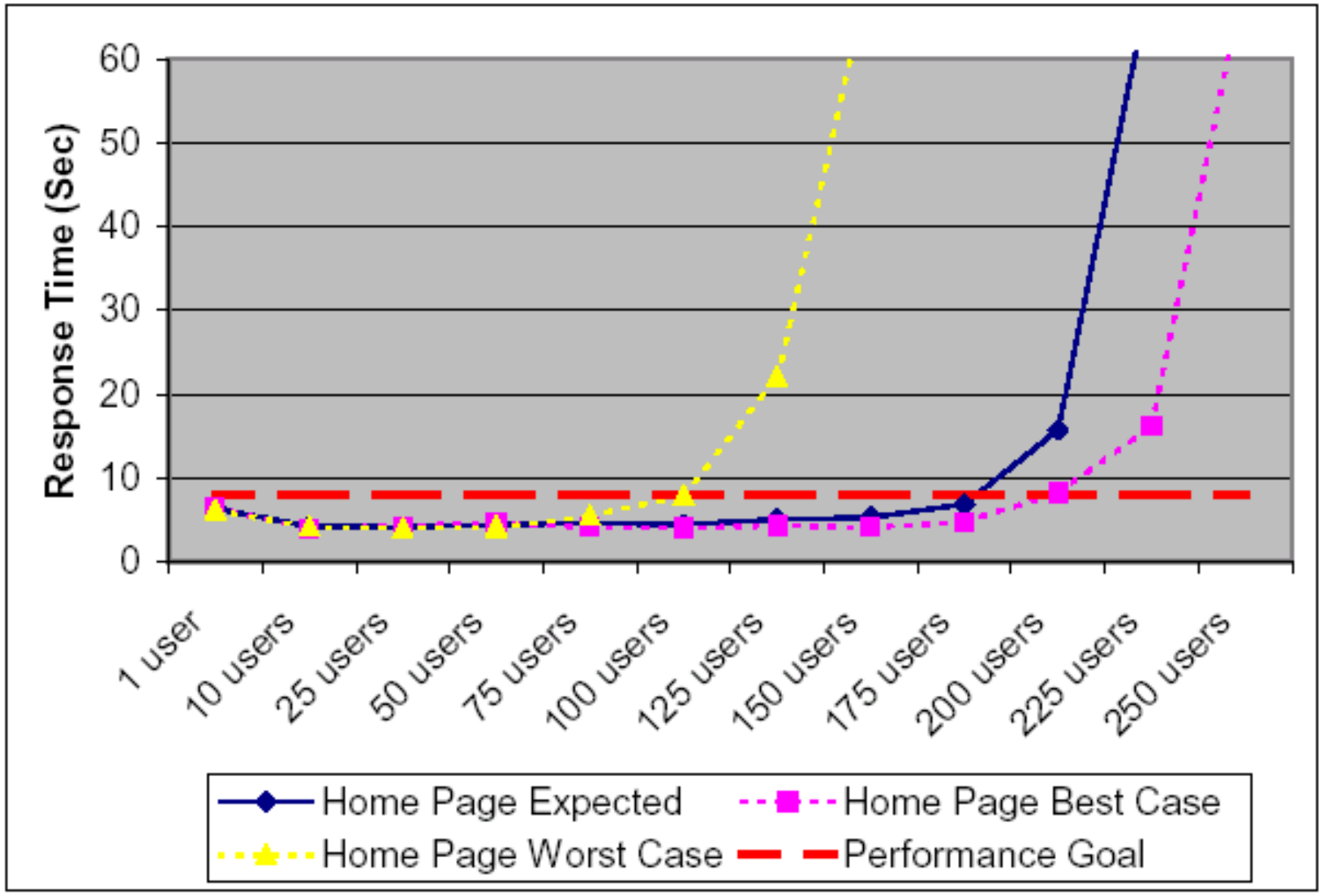


Report

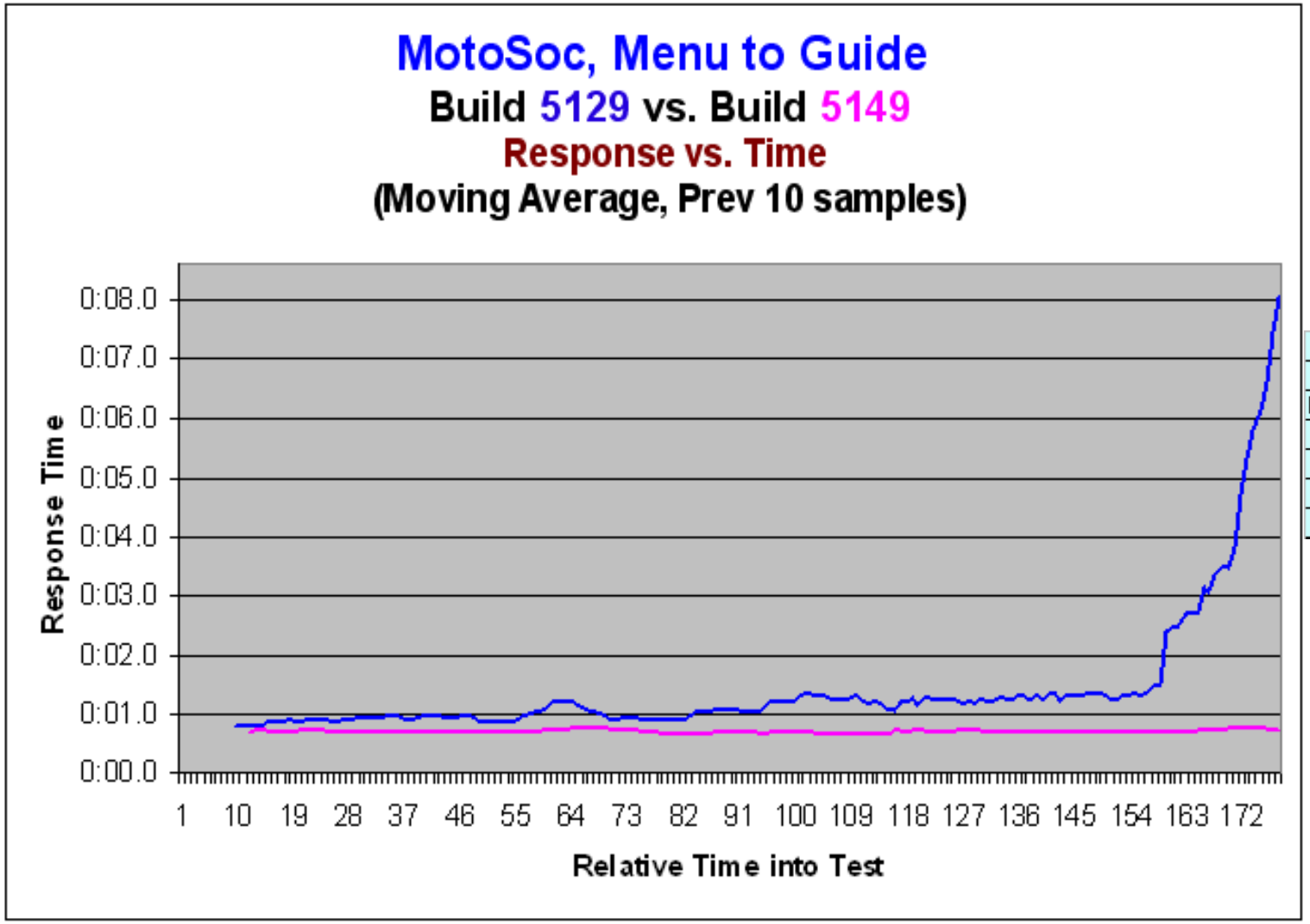


	1 user	10 users	25 users	50 users	75 users	100 users	125 users	150 users	175 users	200 users
HomePage	6.38	4.25	4.14	4.48	4.59	4.48	5.07	5.33	6.89	15.70
Page 1	4.77	4.01	3.67	3.60	3.81	3.71	4.19	4.71	4.85	12.10

Report



Report



	Bld. 5129	Bld. 5149
Avg:	0:01.60	0:00.70
Median:	0:00.98	0:00.66
90th:	0:01.97	0:00.77
STD:	0:01.90	0:00.08
Min:	0:00.66	0:00.07
Max:	0:10.00	0:00.88

Iterate

Iterate

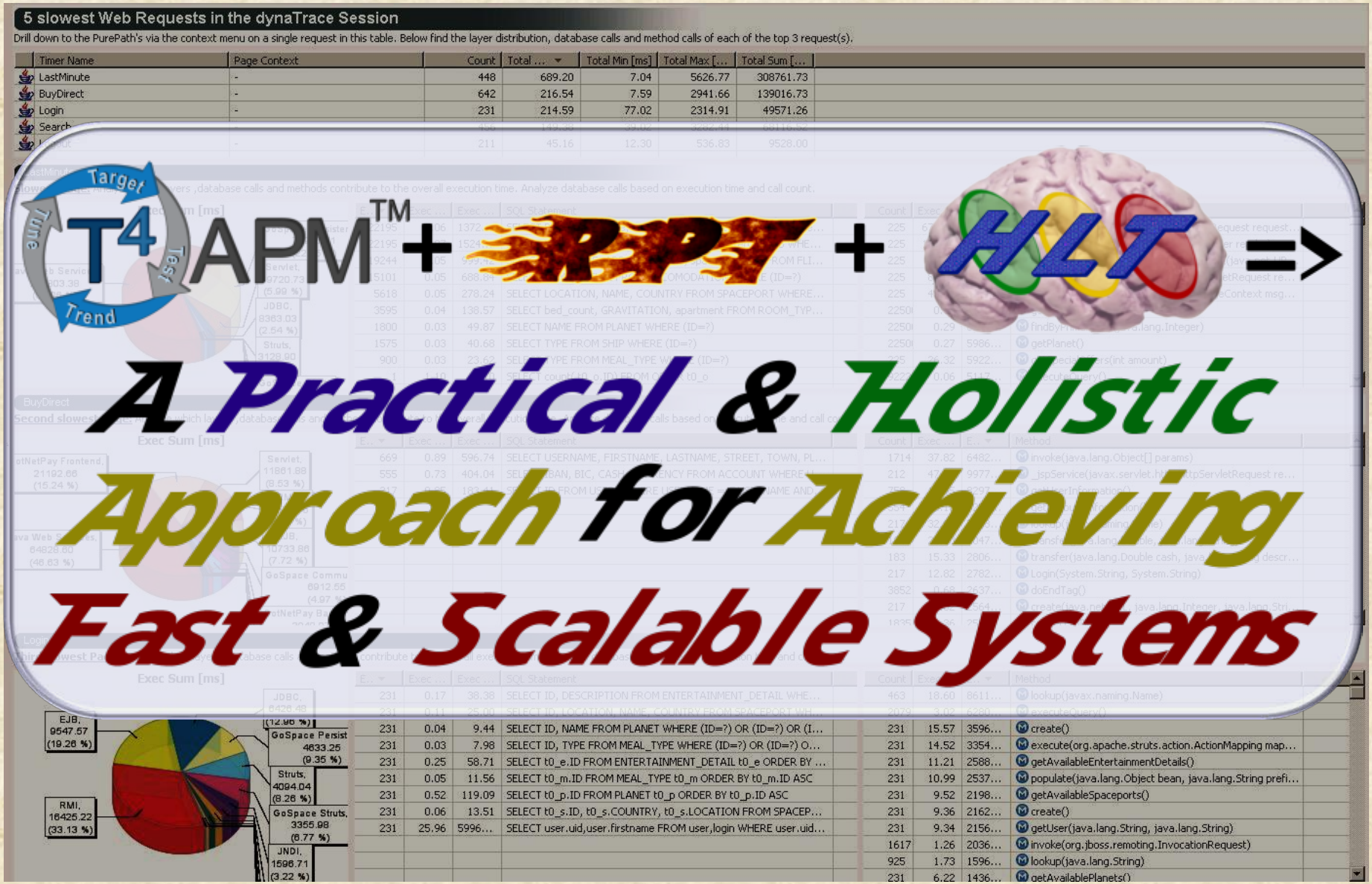
- Don't confuse "Delivery" with "Done"
- You will never have enough data (statistically), even if you already have too much (to parse effectively).
- Ask "Rut or Groove".
- Don't let complacency be your guide.
- If you run out of new ideas, take old ideas to new extremes.
- Above all else ask:

*"What test that I can do **right now**, will add the most **informational value** to the project?"*

Load Testing Principles

Context	Project context is central to successful performance testing.
Criteria	Business, project, system, & user success criteria.
Design	Identify system usage, and key metrics; plan and design tests.
Instrument	Install and prepare environment, tools, & resource monitors.
Script	Script the performance tests as designed.
Execute	Run and monitor tests. Validate tests, test data, and results.
Analyze	Analyze the data individually and as a cross-functional team.
Report	Consolidate and share results, customized by audience.
Iterate	"Lather, rinse, repeat" as necessary.

The Bottom Line



Contact Info

Scott Barber

Chief Technologist

PerfTestPlus, Inc

E-mail:

sbarber@perftestplus.com

Web Site:

www.PerfTestPlus.com

Blog:

scott-barber.blogspot.com

Twitter:

@sbarber

Review & Questions

Did we learn anything?