Performance Testing Software Systems:

A Heuristic Approach

Derived from:

Microsoft patterns & practices

Performance Testing Guidance for Web Applications

By: J.D. Meier, Carlos Farre, Prashant Bansode, Scott Barber, Dennis Rea

http://www.codeplex.com/PerfTestingGuide

Scott Barber

Chief Technologist

PerfTestPlus, Inc.

Performance Testing Software Systems

Scott Barber, (A.K.A. *"The Perf Guy"*)
sbarber@perftestplus.com

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

Vice President & Executive Director,
Association for Software Testing
www.associationforsoftwaretesting.org

Contributing Author & Expert Advisor
Performance Testing Guidance for Web Applications
www.codeplex.com/PerfTestingGuide

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, ©1995-2007 Satisfice, Inc.)

Just-In-Time Testing (Robert Sabourin, ©1998-2007 Amibug, Inc.)

I Assume That You:

Test software performance or manage someone(s) who does.

Have at least some control over the design of your tests and some time to create new tests.

Have at least some influence over your test environment.

Are worried that your test process is spending too much time and resources on things that aren't important AND/OR

Are worried that your test process doesn't leave enough time and resources to determine what IS important.

Believe that good testing requires thinking.

Test under uncertainty, resource limitations and time pressure.

Have a major goal to find important problems quickly.

Want to get very good at testing software performance.

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

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

--Scott Barber

Instructional Methods That I Use

The Class Presents My Editorial Opinions: I do not make appeals to authority; I speak only from my experiences, and I appeal to your experience and intelligence.

Not All Slides Will be Discussed: There is *much* more material here than I can cover in detail, so I may skip some of it. (If you want me to go back to something that I skipped, just ask.)

I Need to Hear from You: You control what you think and do, so I encourage your *questions about* and *challenges to* the lecture. (Talk to me during the break, too.)

If You Want Specifics, Bring Specifics: I invite you to bring real examples of testing problems and test documents to class. (I am happy to show you how I would work through them.)

The Exercises are the Most Important Part: I sometimes use *immersive socratic exercises* that are designed to fool you if you don't ask questions. I usually do not provide all the information you need. *Asking questions is a fundamental testing skill!*

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

Instructional Methods That I Use

I am likely to push you

If I call on you,
and you don't want to be put on the spot,
just say "**Pass!**"

But you can push back

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

What Not to Expect From Me

Untested theory.

Marketing fluff.

Pulled punches to protect the guilty.

The “One True Answer” to anything.

Every concept to apply, precisely as presented, to every context.

Over simplifications without acknowledgement.

A disimpassioned, boring instructor!

Primary Goal of this Course

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

Secondary Goal of this Course

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

“Let’s face the truth, performance testing

***IS* rocket science.”**

--Dawn Haynes

Mnemonic

Mnemonics make difficult to remember information, less difficult to remember.

adjective:

"assisting or intended to assist the memory."

noun:

"thing intended to assist the memory, as a verse or formula."

"Mnemonics Neatly Eliminate Man's Only Nemesis – Insufficient Cerebral Storage"
An acrostic mnemonic to remember how to spell "mnemonics".
- http://www.mnemonic-device.eu

Mnemonics in this Course

Were created by Scott Barber to help Scott organize and remember stuff.

Have been, and will be, revised when revisions help Scott remember stuff better, or remember better stuff.

Have been used and liked by some people other than Scott.

Have proven to be “not so memorable” for some people.

If these mnemonics don’t work for you, create your own!!

Heuristics

Heuristics bring useful structure to problem-solving skill.

adjective:

"serving to discover."

noun:

"a fallible method for solving a problem or making a decision."

"Heuristic reasoning is not regarded as final and strict but as provisional and plausible only, whose purpose is to discover the solution to the present problem."
- George Polya, *How to Solve It*

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

Types of Heuristics

Guideword Heuristics: Words or labels that help you access the full spectrum of your knowledge and experience as you analyze something.

Trigger Heuristics: Ideas associated with an event or condition that help you recognize when it may be time to take an action or think a particular way. Like an alarm clock for your mind.

Subtitle Heuristics: Help you reframe an idea so you can see alternatives and bring out assumptions during a conversation.

Heuristic Model: A representation of an idea, object, or system that helps you explore, understand, or control it.

Heuristic Procedure or Rule: A plan of action that may help solve a class of problems.

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

Some Everyday Heuristics

It's dangerous to drink and drive.

A bird in hand is worth two in the bush.

Nothing ventured, nothing gained.

Sometimes people stash their passwords near their computers. Try looking there.

Stores are open later during the Holidays.

If your computer is behaving strangely, try rebooting. If ~~it's very strange, reinstall Windows~~.

If it's a genuinely important task, your boss will follow-up, otherwise, you can ignore it.

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

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.

Heuristics in this Course

Were created by Scott to help Scott organize and remember stuff.

Are fallible (ask Scott about times when they have failed him).

Are incomplete (ask Scott about times when he used these and still missed “stuff”).

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)!!

Principles ≠ Process

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

In my experience, successful performance testing projects involve at least active decisions related to each of the core principles in this course.

Core principles are neither exclusive, nor sequential. They go by many different names, have varying priorities, and may be implicit or explicit.

Core principles are not in themselves an approach or process.

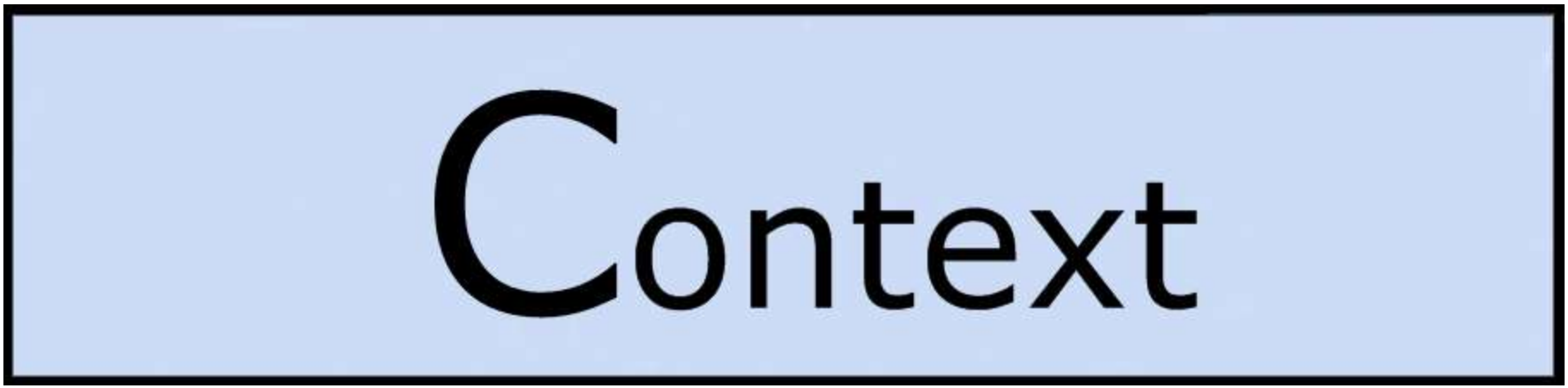
Core principles represent a foundation upon which to build a process or approach based on the context of your project.

Performance Testing Principles:

(A mnemonic of guideword heuristics)

Performance 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.
Install	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.



When assessing project context, I

(Another mnemonic of guideword heuristics)

Why do we test software system performance?

To determine compliance with requirements?

To evaluate release readiness?

To assess user satisfaction?

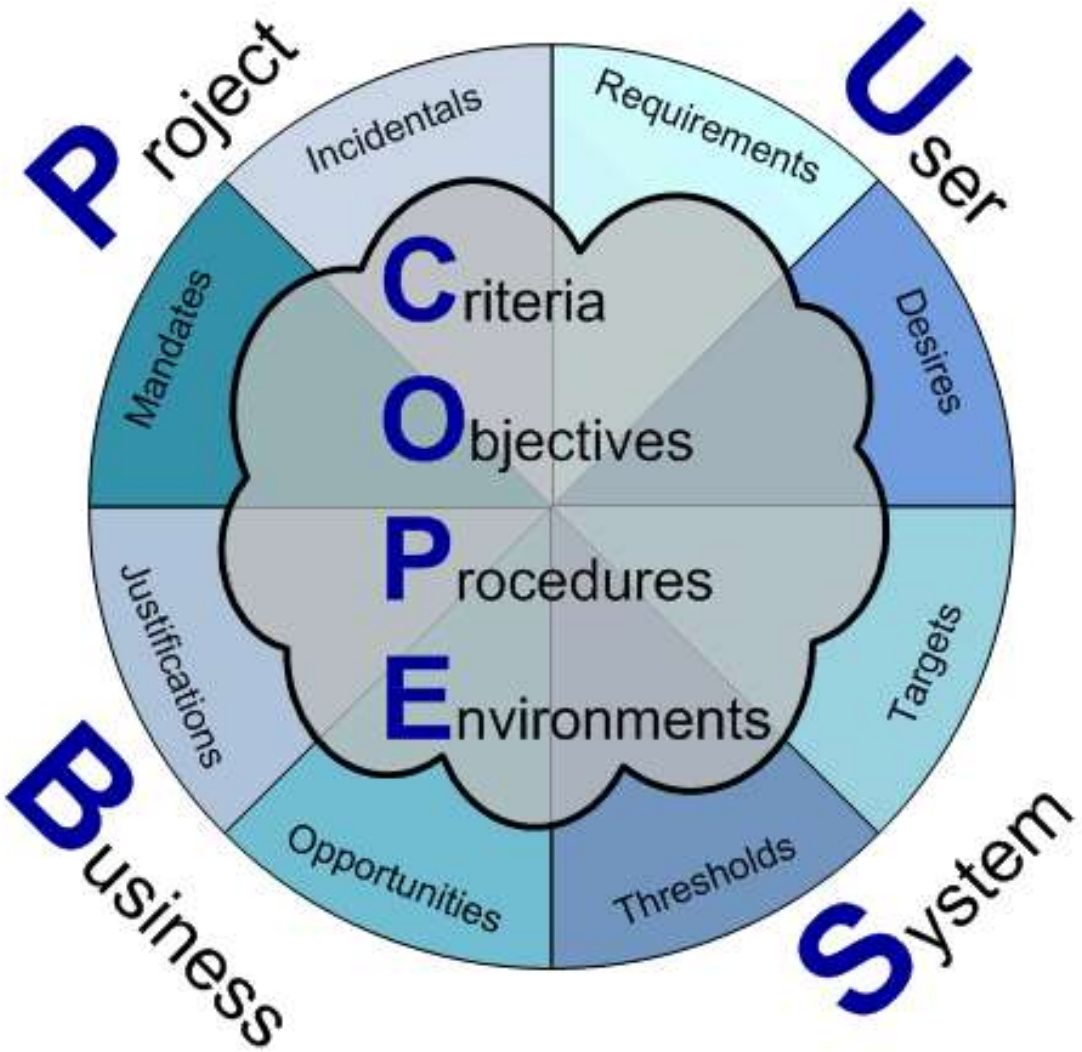
To assist in performance tuning?

To estimate capacity?

To validate assumptions?

To generate marketing statements?

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.

Context

Instructions:

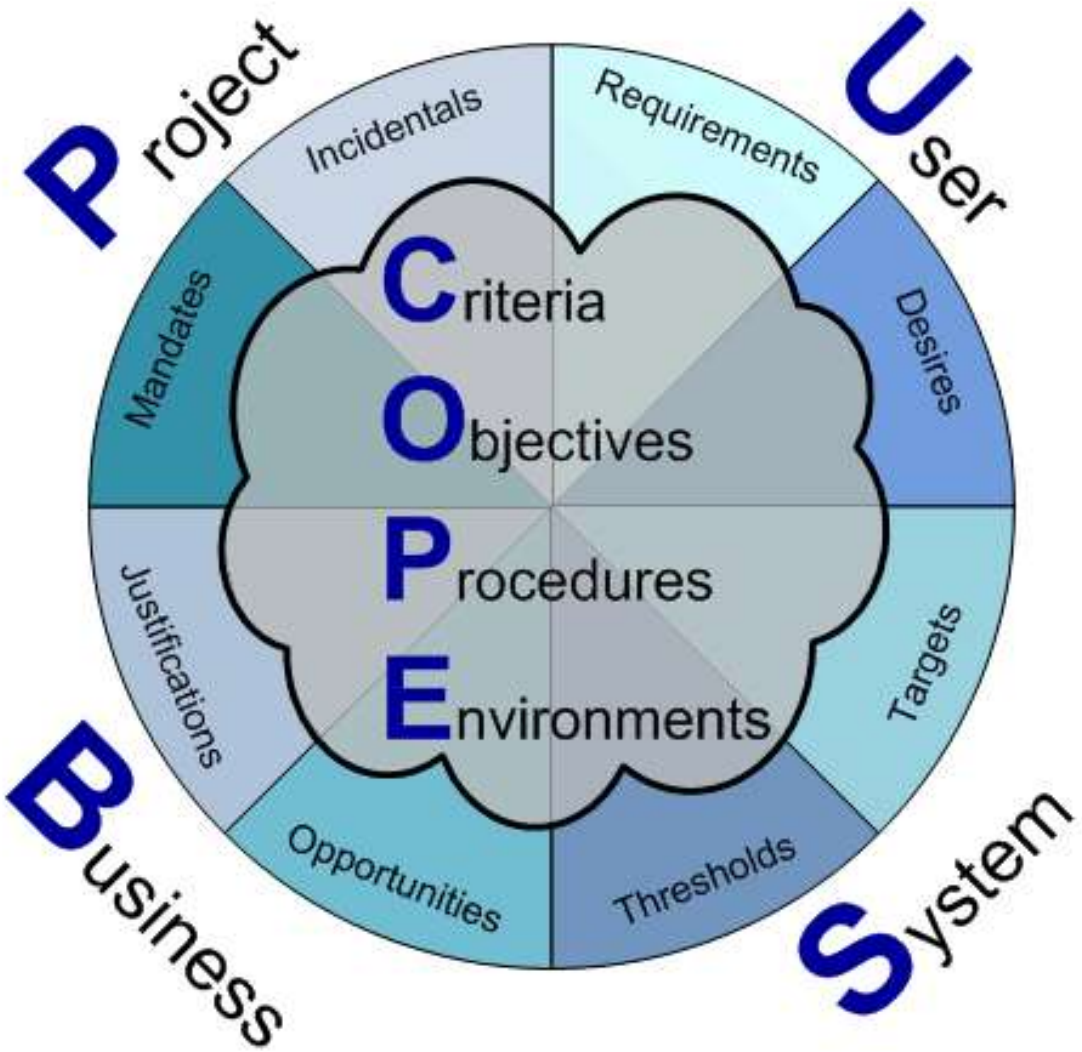
Assemble into groups of 3-5 people (4 is ideal) that you don't normally work with.

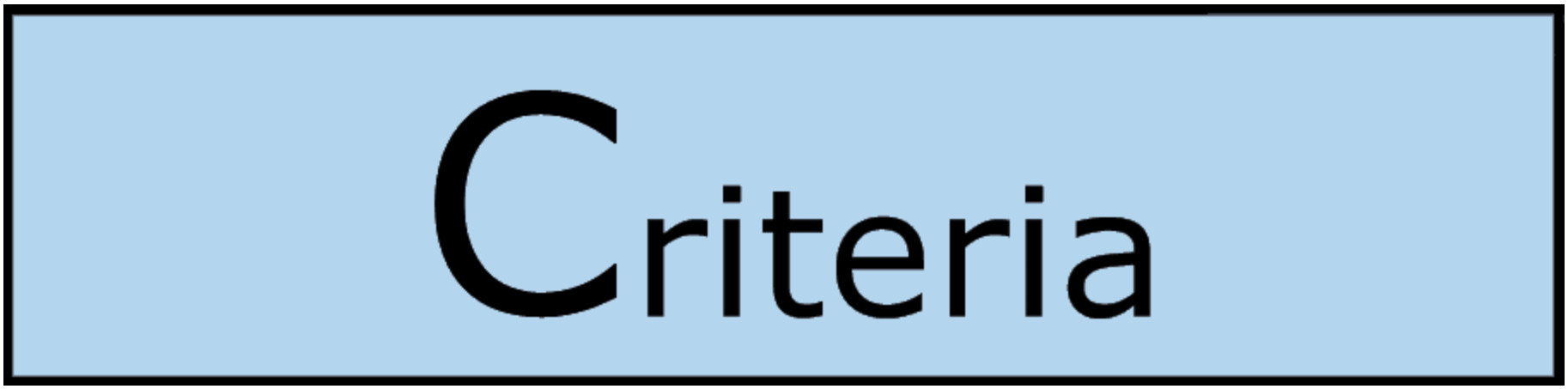
Use COPE in PUBS to describe your current or most recent performance testing project to one another.

Pick the most interesting and jot down some notes about its context using COPE in PUBS.

Be prepared to brief the class on the context of the project you chose AND be prepared to answer questions about the contexts of your teammates.

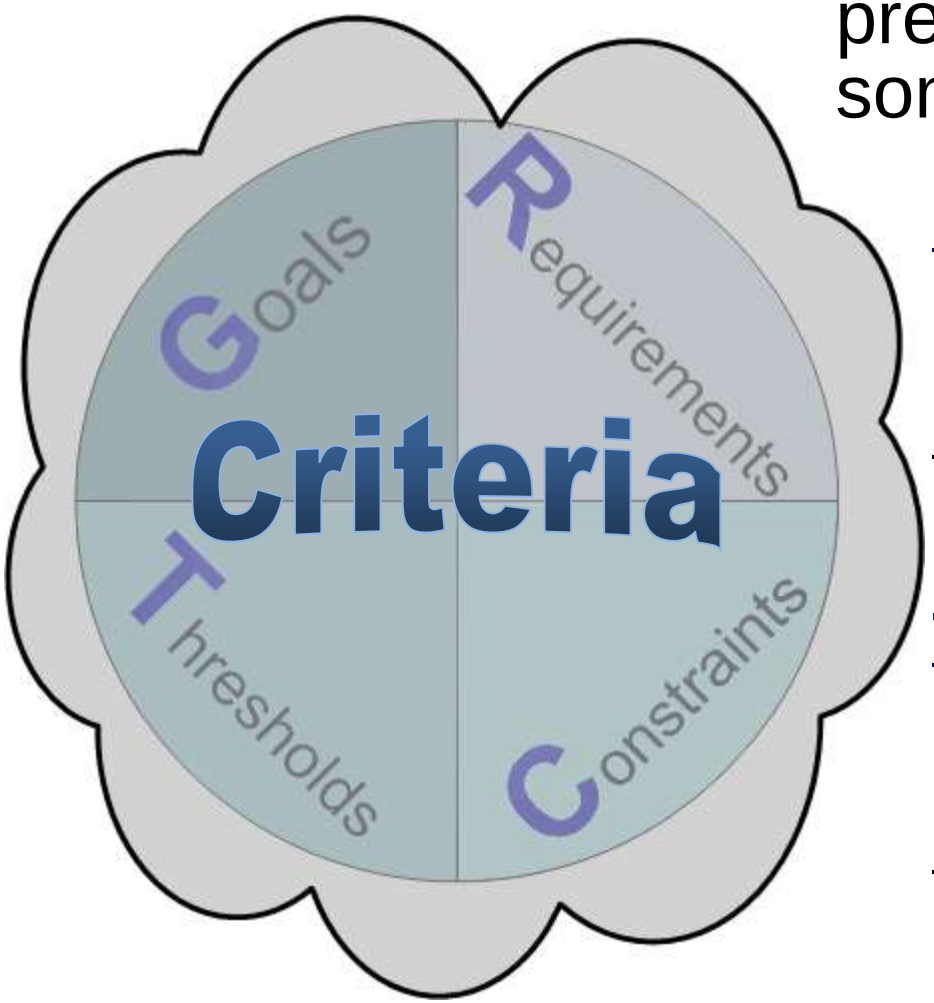
Context





Criteria

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



Goals: Soft Boundaries
(User Satisfaction)

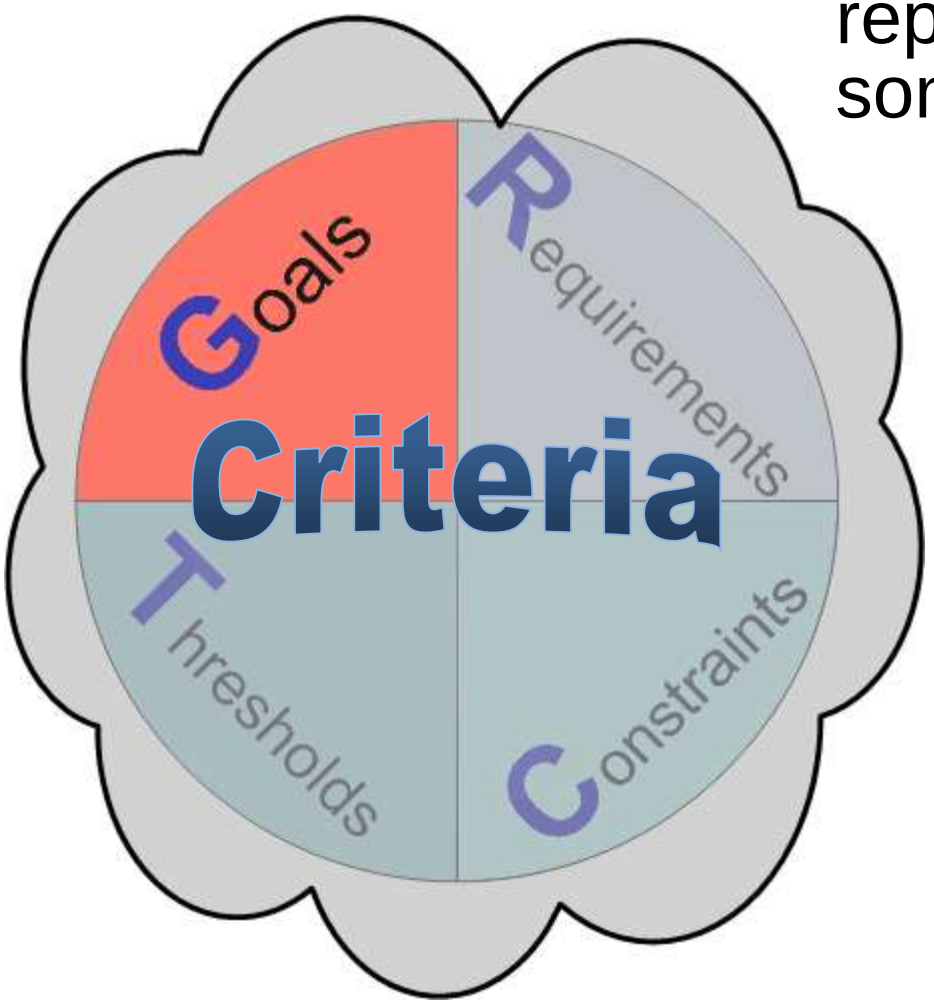
Requirements: Firm Boundaries
(Business or Legal)

Thresholds: Hard Boundaries
(Laws of Physics)

Constraints: Arbitrary Boundaries
(Budget or Timeline)

Criteria

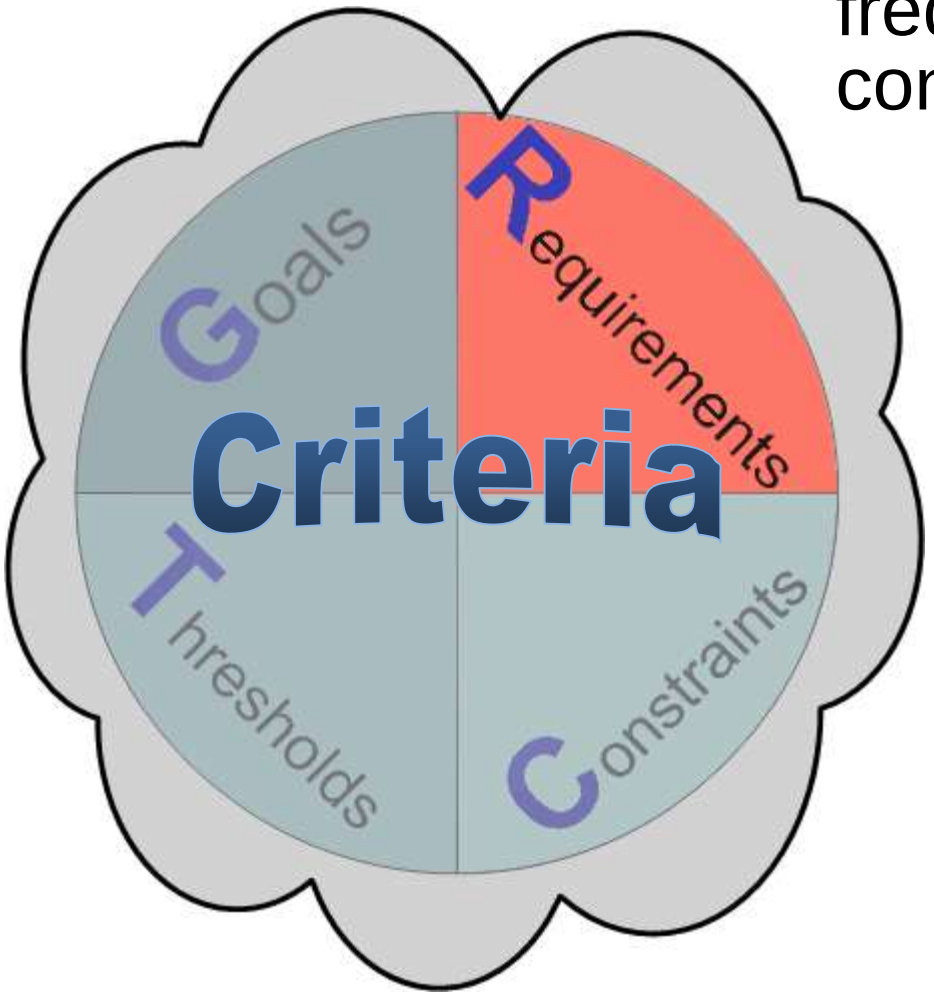
Performance Goals are *soft boundaries* typically representing the opinion of someone that matters.



- Typically target whole-system performance characteristics
- Are often “unsubstantiated opinions”
- Are often unattainable
- Must be well qualified, but can be loosely quantified

Criteria

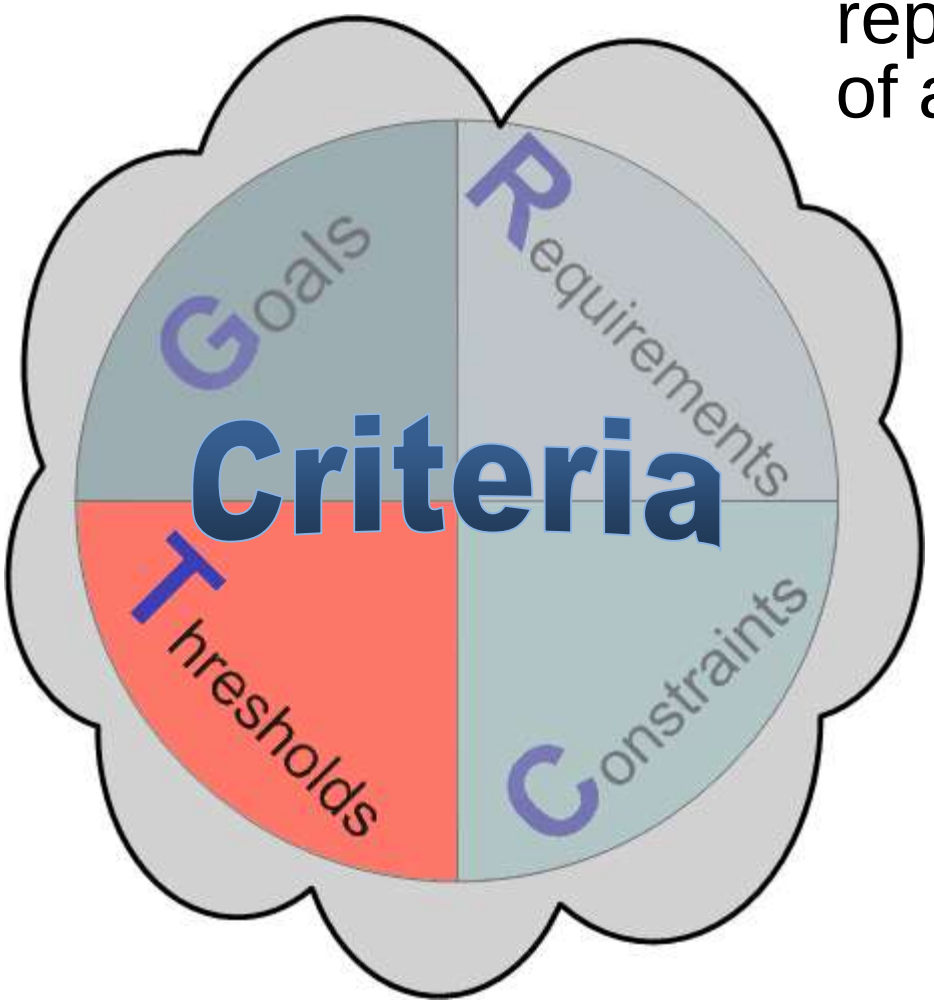
Performance Requirements are *firm boundaries* frequently derived from contracts (that matter).



- Are actually required to pass to go live
- Are often externally dictated
- Are often continually monitored in production
- Are typically legally enforceable
- Must be both well qualified and quantified

Criteria

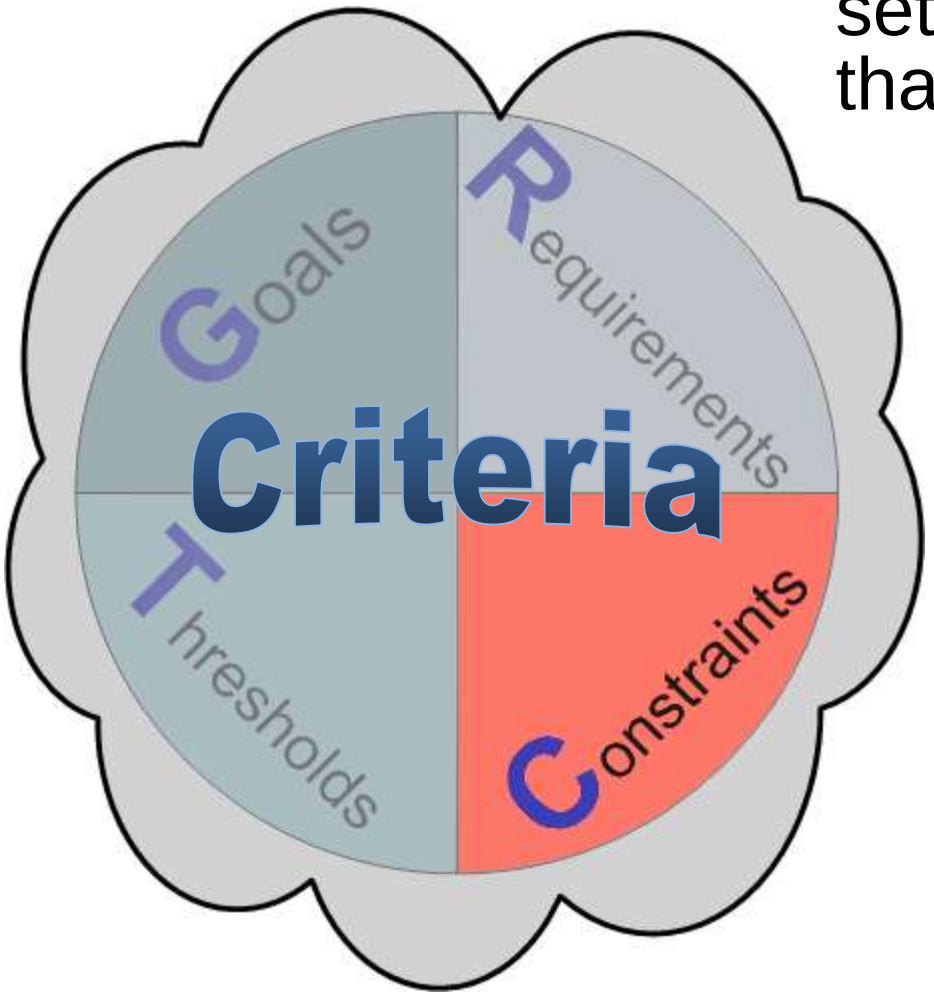
Performance Thresholds are *hard boundaries* that represent physical properties of a system (that matters).



- Are the maximum acceptable values for component-level resources of interest
- Based on published hardware or software performance recommendations or direct observation
- Are practically non-negotiable without an environmental change

Criteria

Performance Constraints are *arbitrary boundaries* set or presumed by someone that matters (but likely doesn't get "it").



- Boundaries imposed by people, traditions, and/or assumptions
- Are sometimes exceedingly difficult to challenge
- Are almost always worth questioning if they jeopardize the project

Criteria

Context + Criteria =>

Performance Testing Objectives

Performance 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

How do we know if we’re meeting our objectives?

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.

“..it works”

really means...

“...it appeared at least once to meet some requirement to some degree.”

One or more successes!

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

Criteria

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.

All Oracles Are Heuristic

We often do not have oracles that establish a definite correct or incorrect result, in advance.
That's why we use abductive inference.

No single oracle can tell us whether a program (or a feature) is working correctly at all times and in all circumstances.
That's why we use a variety of oracles.

Any program that looks like it's working, to you, may in fact be failing in some way that happens to fool all of your oracles.
That's why we proceed with humility and critical thinking.

You (the tester) can't know the deep truth about any result.
That's why we report whatever seems *likely* to be a bug.

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

Coping With Difficult Oracle Problems

Ignore the Problem

Ask "so what?" Maybe the value of the information doesn't justify the cost.

Simplify the Problem

Ask for testability. It usually doesn't happen by accident.

Built-in oracle. Internal error detection and handling.

Lower the standards. You may be using an unreasonable standard of correctness or goodness.

Shift the Problem

Parallel testing. Compare with another instance of a comparable algorithm.

Live oracle. Find an expert who can tell if the output is correct.

Reverse the function. (e.g. 2 x 2 = 4, then 4/2 = 2)

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

Coping With Difficult Oracle Problems

Divide and Conquer the Problem

Spot check. Perform a detailed inspection on one instance out of a set of outputs.

Blink test. Compare or review overwhelming batches of data for patterns that stand out.

Easy input. Use input for which the output is easy to analyze.

Easy output. Some output may be obviously wrong, regardless of input.

Unit test first. Gain confidence in the pieces that make the whole.

Test incrementally. Gain confidence by testing over a period of time.

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

Criteria

Instructions:

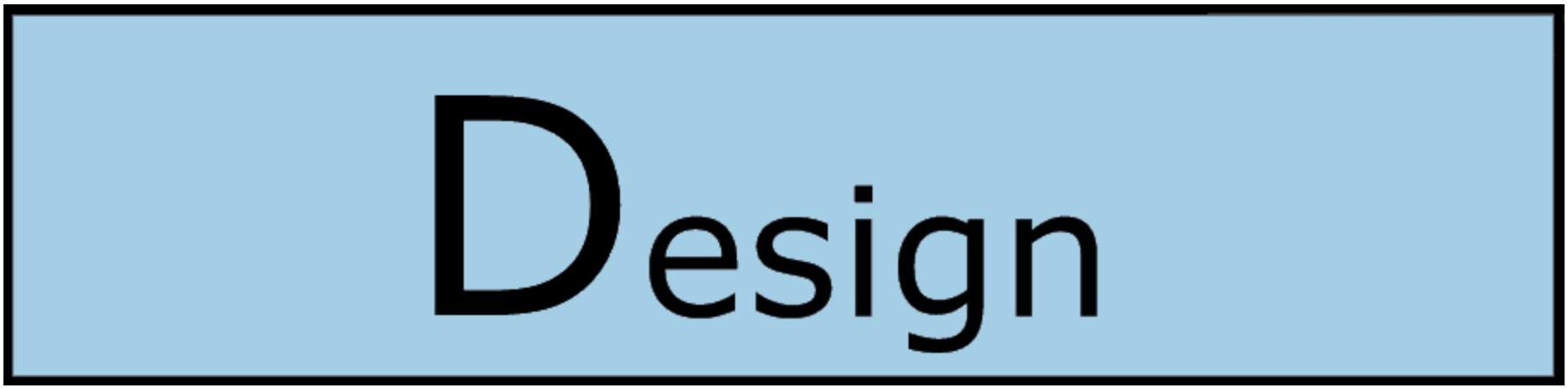
Reassemble into your group.

Identify at least 3 performance criteria for your project of each type. (Goals, Requirements, Thresholds, and Constraints)

Based on your criteria, identify the top 5 performance testing objectives for your project.

Identify your oracles for the top 5 performance testing objectives.

Be prepared to brief the class on your criteria, objectives, and oracles.



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

Don't be fooled."

--Scott Barber

To help me decide what tests to design, I use

(An acronym of guideword heuristics)

Do I need this test to:

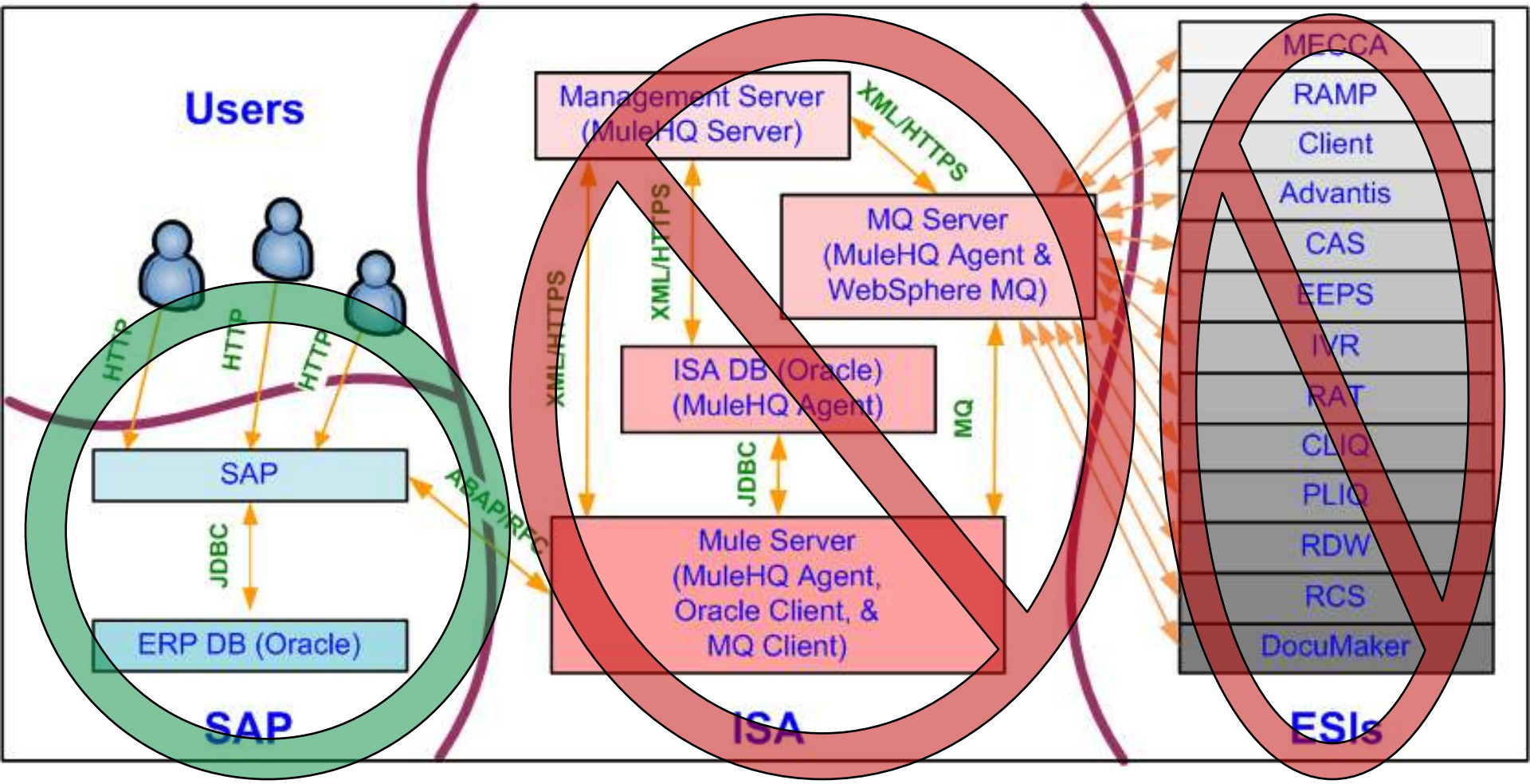
Investigate *or* **Validate/Verify**

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

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

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

Communicating Design



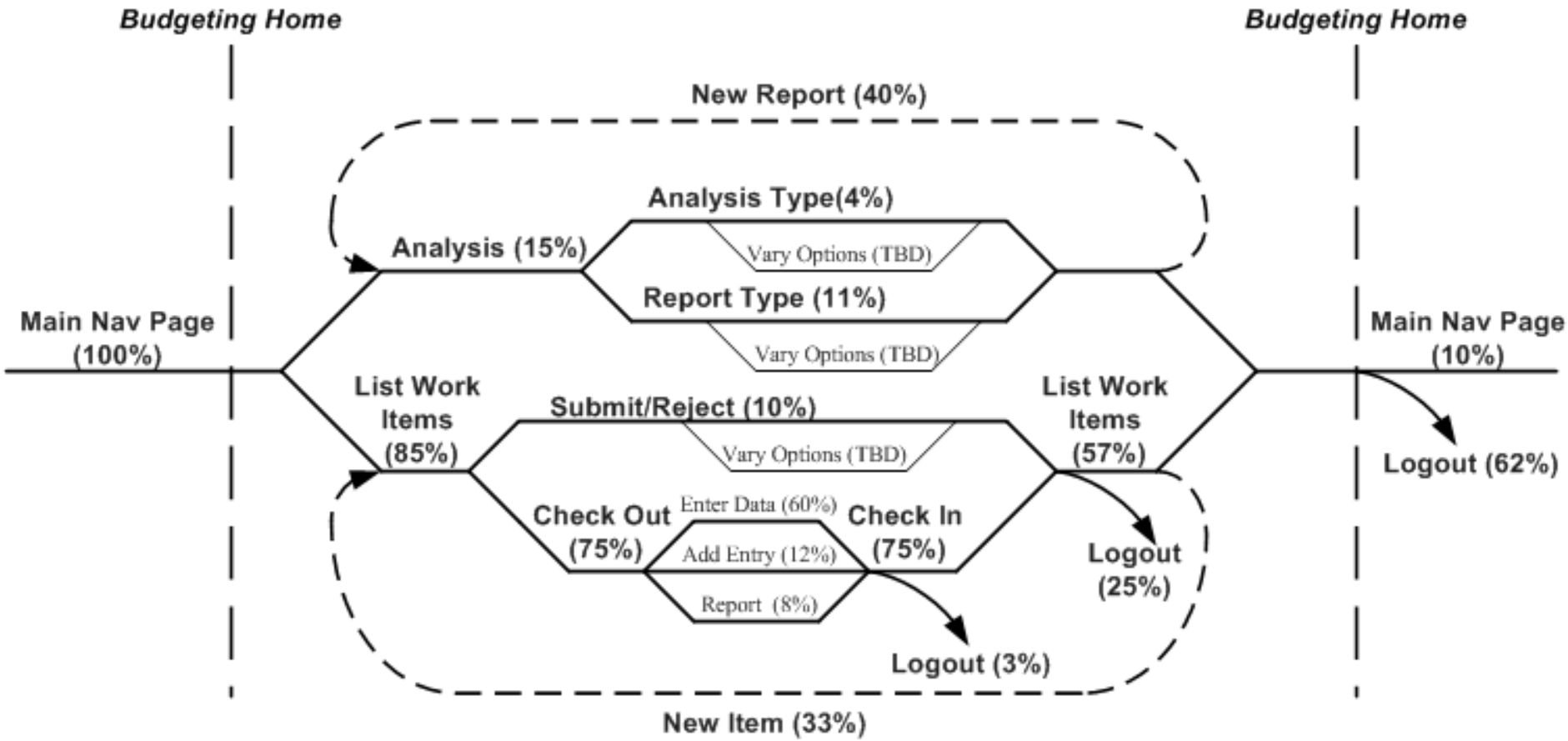
When Building Usage Models, I

(Yet another mnemonic of guideword heuristics)

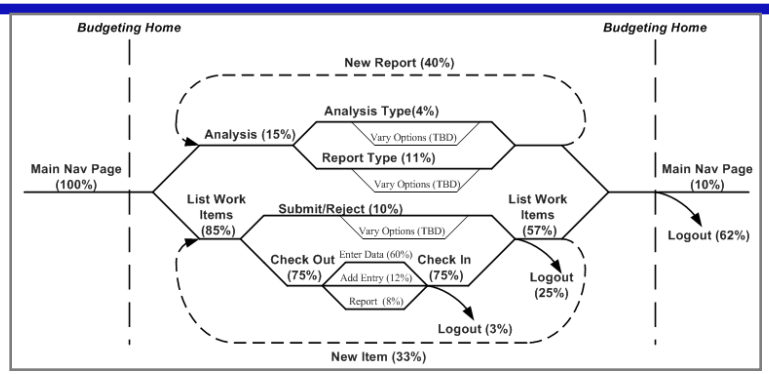
Design

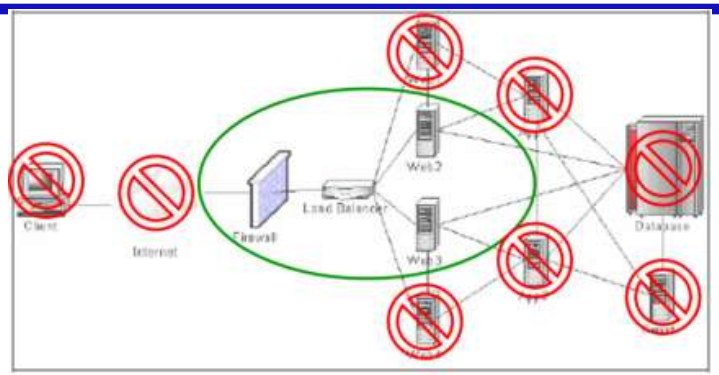
Frequent	Common activities (get from logs)
Intensive	e.g. Resource hogs (get from developers/admins)
Business Critical	Even if these activities are both rare and not risky
Legal or Contract	SLA's, Contracts and other stuff that will get you sued
Obvious	What the users will see and are mostly likely to 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)

Communicating System Usage



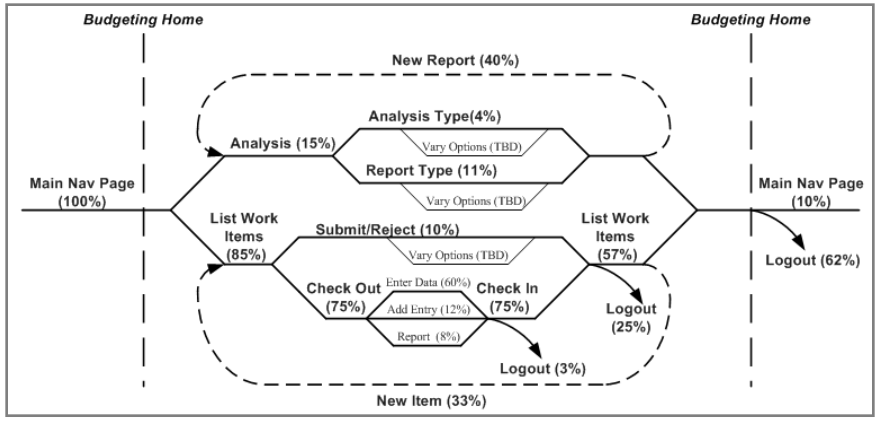
Design

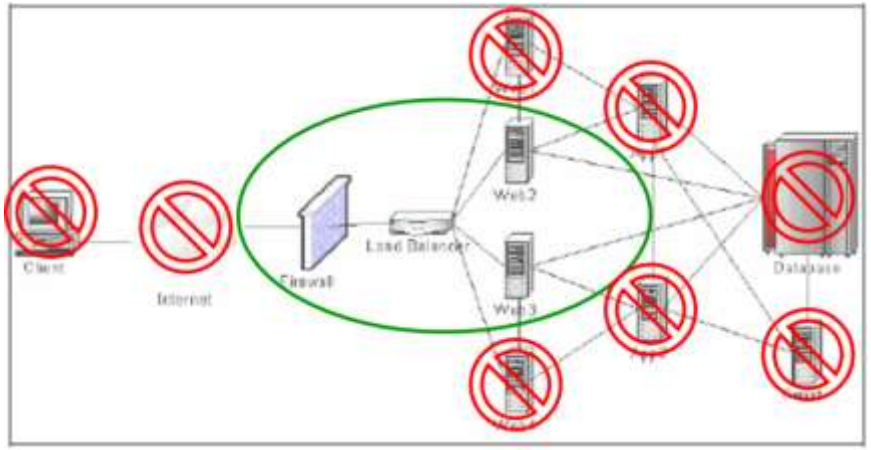




Intent of Investigation:	Collect configuration data for tuning. Collect data to assist in validating existing network.
Prerequisites:	Static prototype deployed on future production hardware.
Tasks:	Determine network bandwidth, validate firewalls & load balancer, evaluate web server settings.
Tools & 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.

Design





Activity

Member Login

Think Time (sec)	Min	Max	Std	Distribution	
	6.0	18.0	2.0	Normal	
Abandon (sec)	**Min**	**Max**	**Std**	**Distribution**	**Event**
	20.0	50.0	N/A	Linear	Repeat
Pass/Fail Condition	If Fail, log data and repeat one time.				

Credentials	Field	Variable Name	Data Description	Data Location
	Username	str_guid	Valid Usernames	Datapool
	Password	str_pwd	Valid Passwords	Datapool

Create Account

Think Time (sec)	Min	Max	Std	Distribution	
	25.0	60.0	8.0	Normal	
Abandon (sec)	**Min**	**Max**	**Std**	**Distribution**	**Event**
	60.0	120.0	N/A	NegExp	Abandon
Pass/Fail Condition	If Fail, log data and abandon user.				

Acct Data	Field	Variable Name	Data Description	Data Location
	Ccard	int_ccard	Valid C-card #'s	File.csv
	Exp_date	int_exp	Valid E-date for C-card	File.csv
	Name	str_cname	Valid Name for C-card	File.csv
	Street	str_street	Valid Street for C-card	File.csv
	City	str_city	Valid City for C-card	File.csv
	State	str_state	Valid State for C-card	File.csv
	Zip	str_zip	Valid zip for C-card	File.csv

Sync Point

Home Page

Type	Parameter(s)
Navigational	None

Condition

In Stock?

Criteria	Resulting Activity(s)
Yes	Purchase
No	Exit

Instructions:

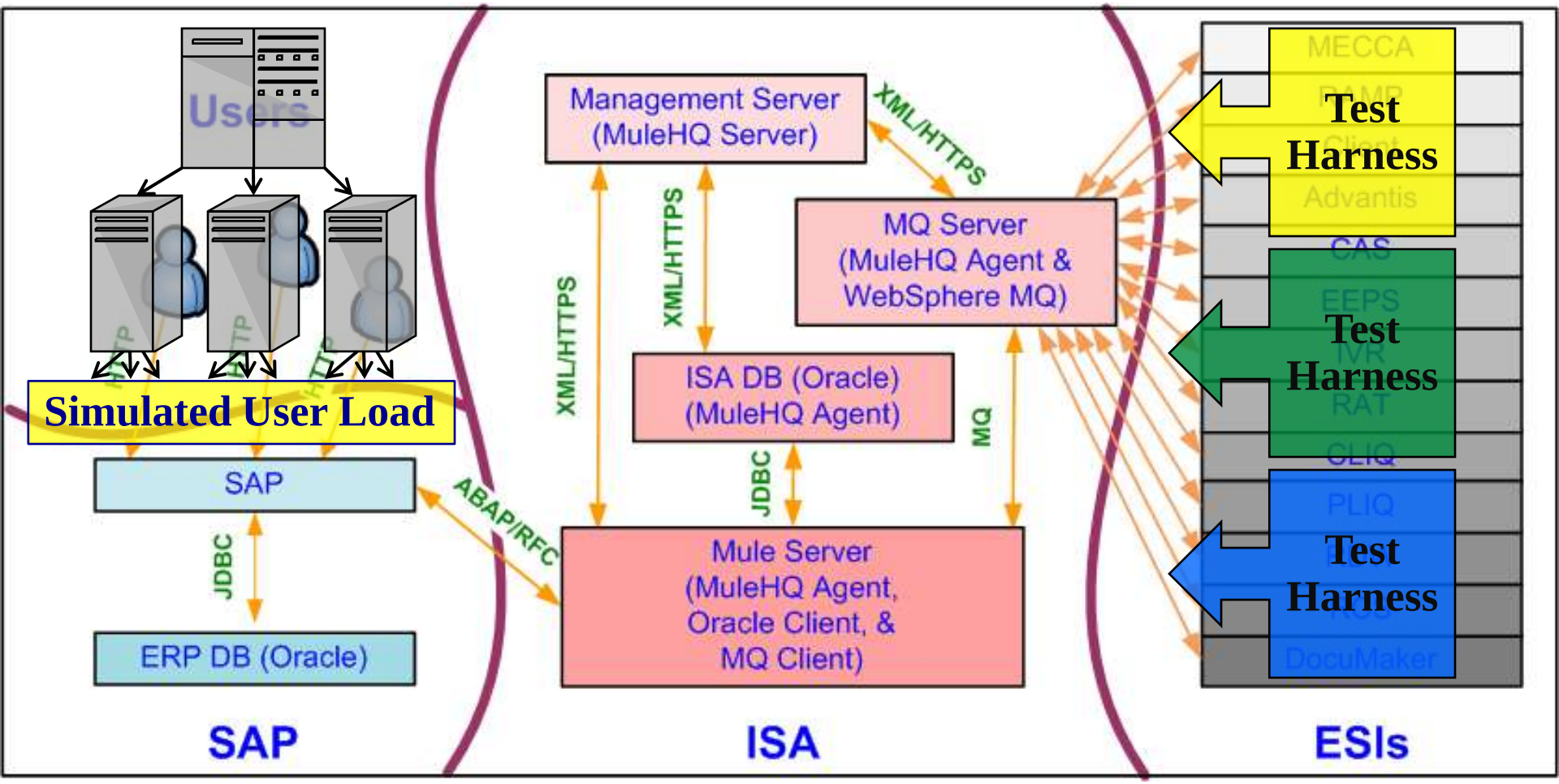
Reassemble into your group.

Referencing your previous work, use IVECTRAS and FIBLOTS to design the top priority performance tests for your team's project.

Revisit your oracles. Will they work as planned? If not, choose new oracles.

Be prepared to brief the class on your tests and your Oracles.

Install



Install

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

"Only performance testing at the conclusion of system or functional testing

is like

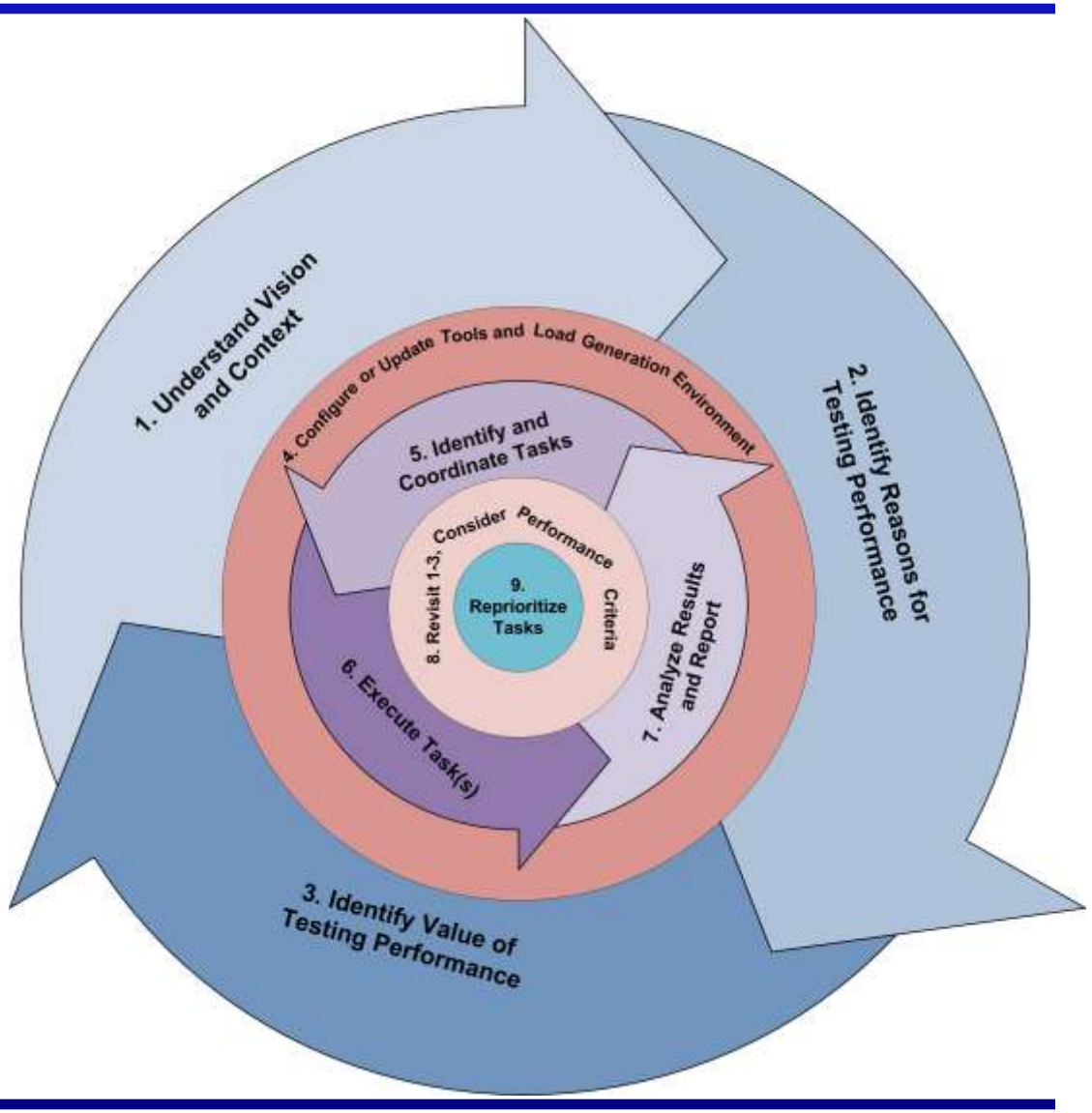
ordering a diagnostic blood test after the patient is dead."

--Scott Barber

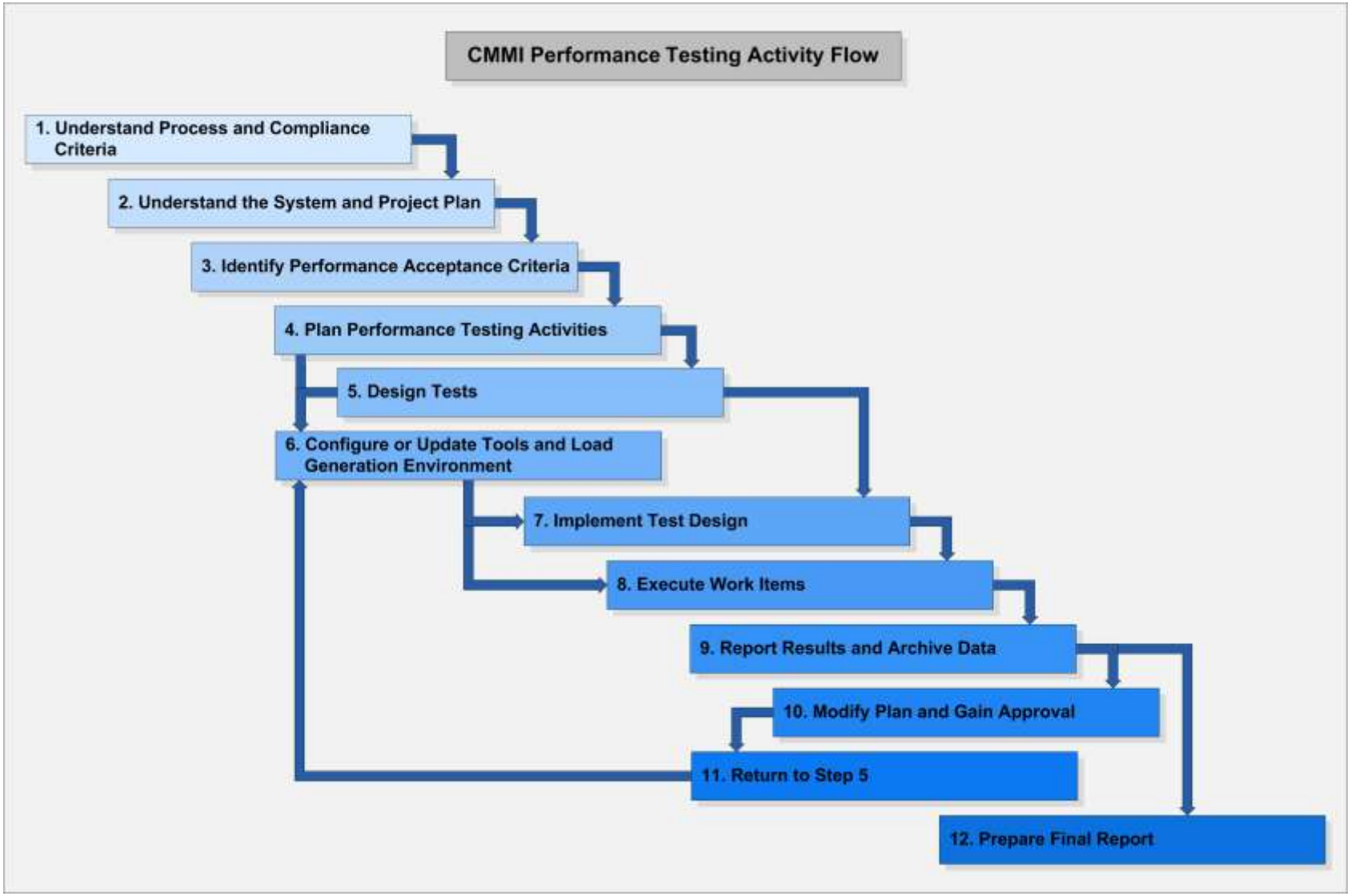
Install

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



Install



Instructions:

Reassemble into your group.

Use the 9 core principles of successful performance testing projects to create an approach or process for your team's project.

Consider how this approach or process will mesh with the overall project approach or process.

Be prepared to brief the class on your process.

“MacGyver is a super-hero,

not

a career path.”

--Scott Barber

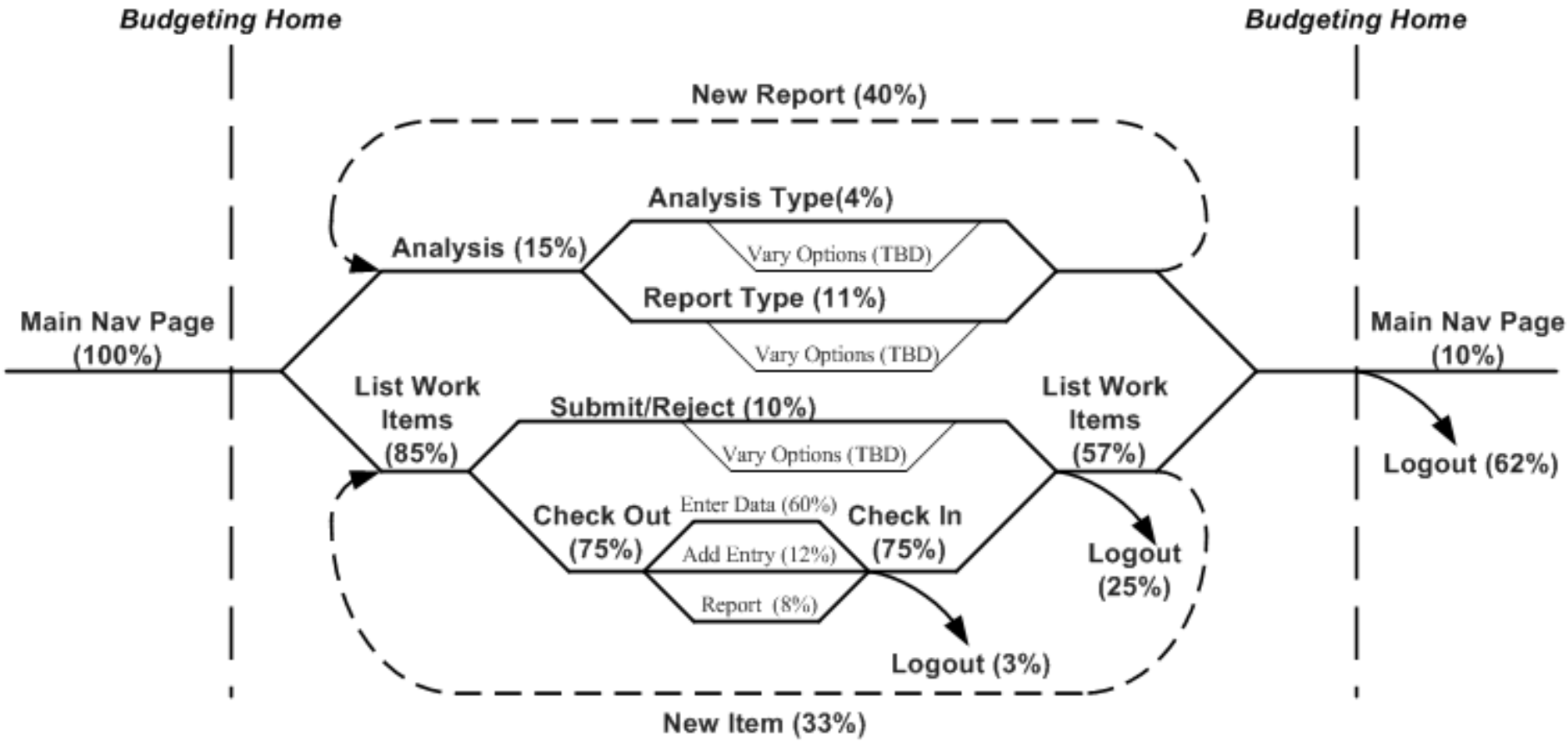
When creating scripts, I try to:

(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

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

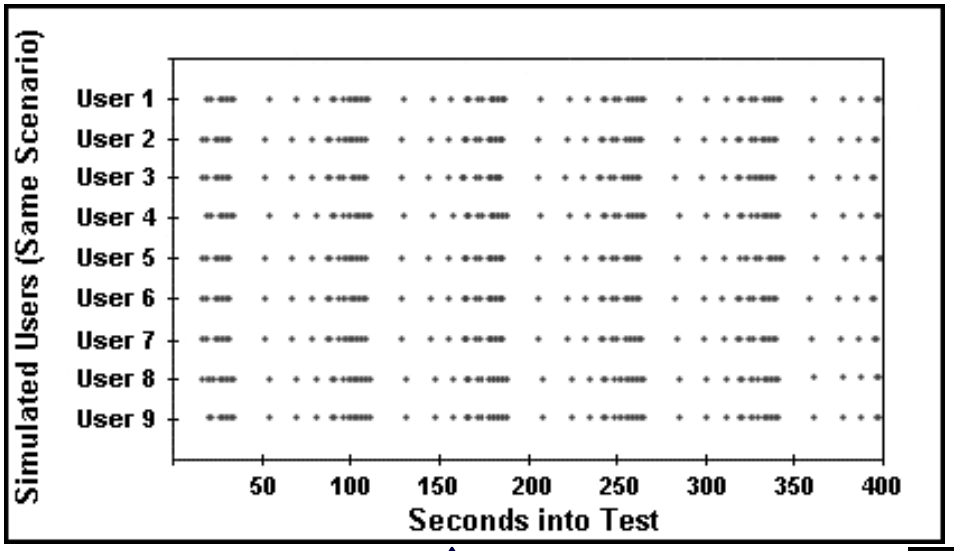
Delays

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.

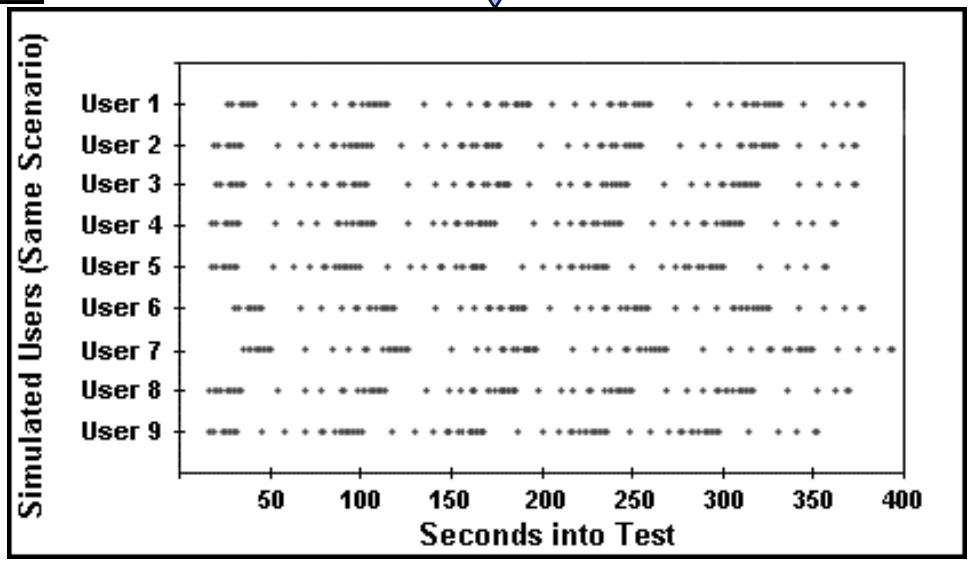
Event Type	Event Name	Type	Min	Max	Std	Req't	Goal
Procedure name:	**Initial Navigation()**						
Timer name:	tmr_home_page	negexp	4	N/A	N/A	8	5
Timer name:	tmr_login	normdist	2	18	4.5	8	5
Timer name:	tmr_page1	linear	5	35	N/A	8	5
Timer name:	tmr_data_entry	negexp	8	N/A	N/A	8	5
Timer name:	tmr_page2	normdist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

Script



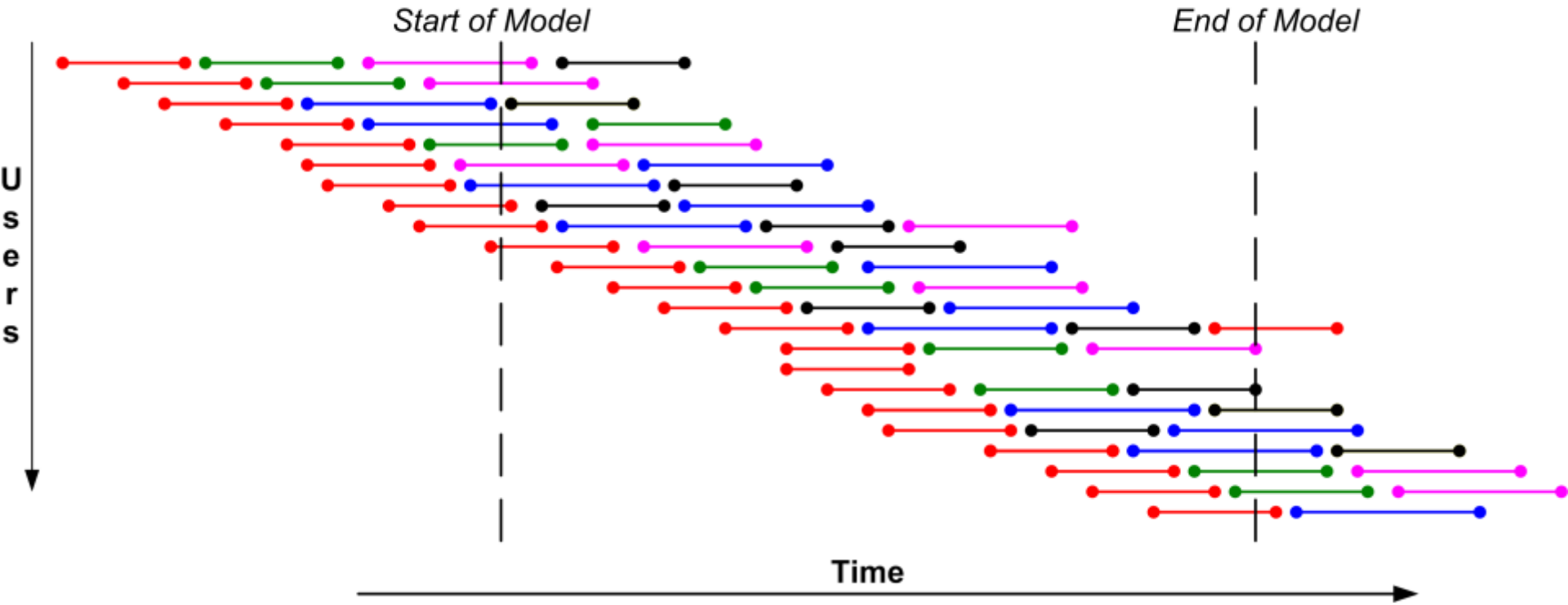
Not Frightening

Frightening

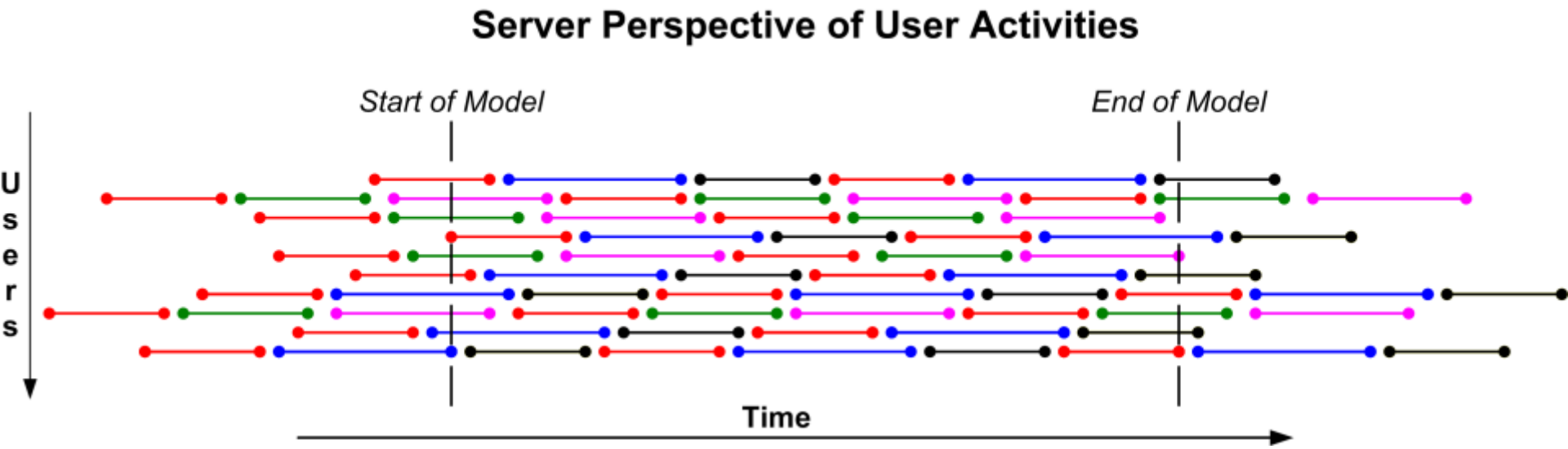


Script

Actual Distribution of User Activities Over Time



Script



Instructions:

Reassemble into your group.

Discuss the most challenging scripting problems you anticipate based on your test design.

Consider alternatives to achieving the same degree of realism while minimizing the scripting challenge.

Be prepared to brief the class on your scripting challenges.

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

(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".

Execution Heuristics:

- "1, 3, 7, 11, More"
- "Best, Expected, Worst"
- "Marching & Resonance"
- "What if Greenspan sneezes?"
- "First user on Monday"
- "UAT under load"
- "If I can't break it, I don't understand it"

Execute

Instructions:

Reassemble into your group.

Spend a few minutes jotting down execution heuristics that may be valuable for your team's project.

Be prepared to describe your heuristics to the class.

"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

When I'm analyzing, I remind myself to:

(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

Methods:

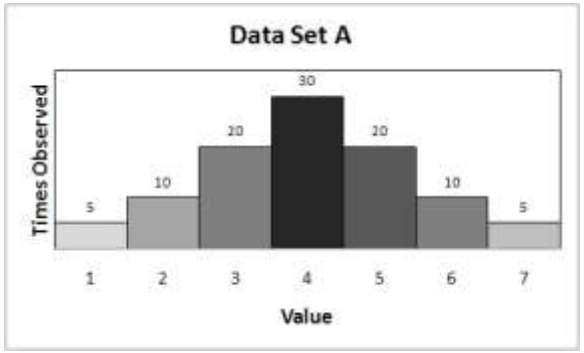
- Blink
- De-Focus & Re-Focus
- Overlay
- Plot
- Bucket
- Look for Odd
- Be Derivative
- Ditch the Digits
- Un-average Averages
- Manual

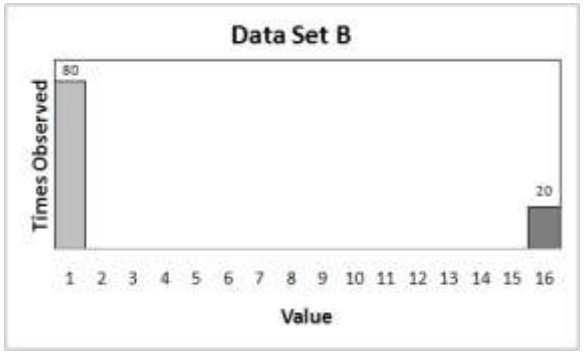
Analyze

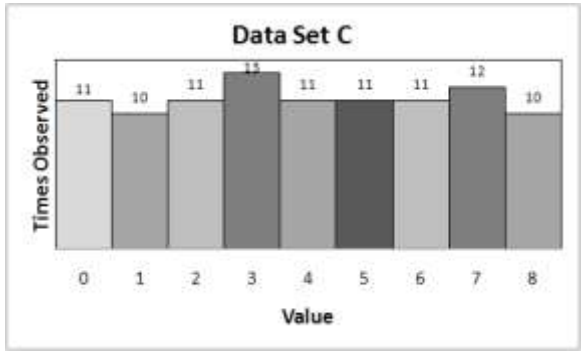
Facts:

- 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?

Instructions:

Reassemble into your group.

Pay attention, I'm going to explain this group of exercises orally.

Be prepared to describe your findings with the class.

"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

I name good reports:

(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.

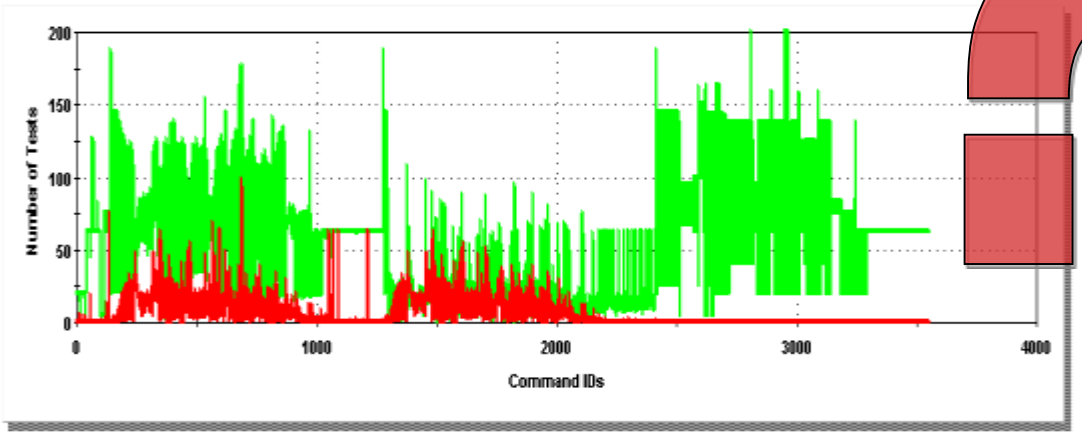
Report

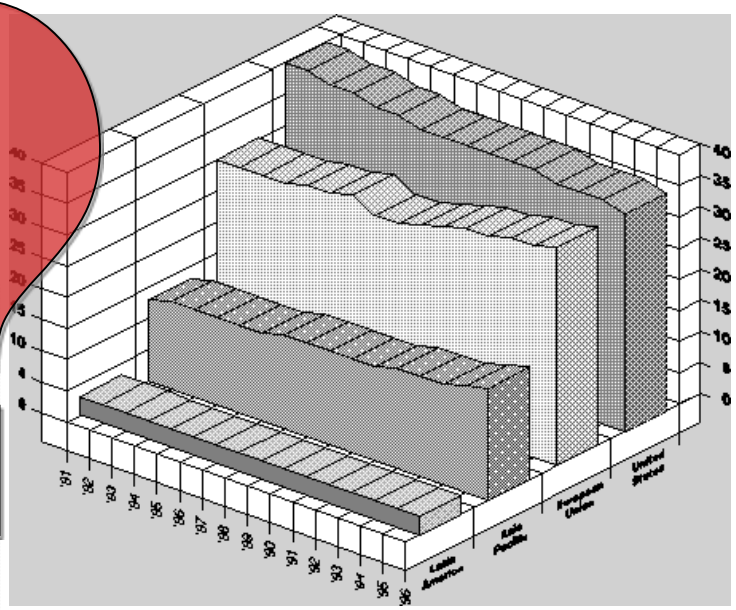
What consumers of reports want:

- 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.

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	10.4
view_faq_window	2.45	2.45	1.47	1.52	1.53	1.66	1.30	6.2
view_faq	0.67	0.67	0.60	0.63	0.58	0.69	0.56	5.4
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.

Inspired by “ET”:

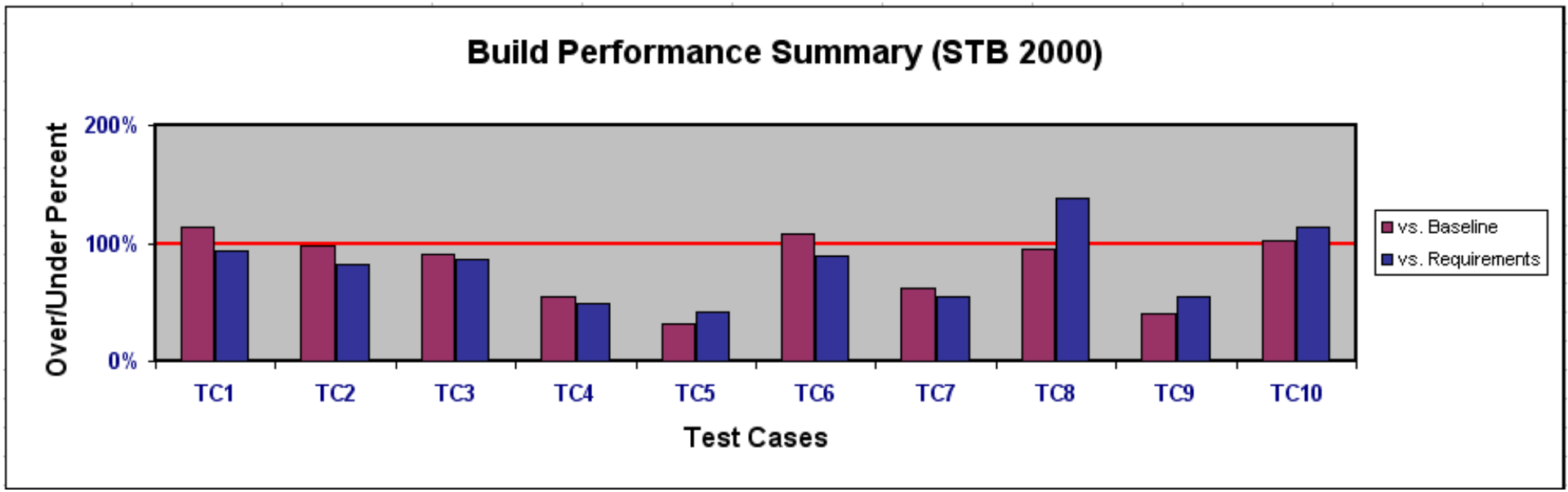
Edward Tufte, Ph.D., Professor Emeritus of political science, computer science and statistics, and graphic design at Yale.

According to ET:

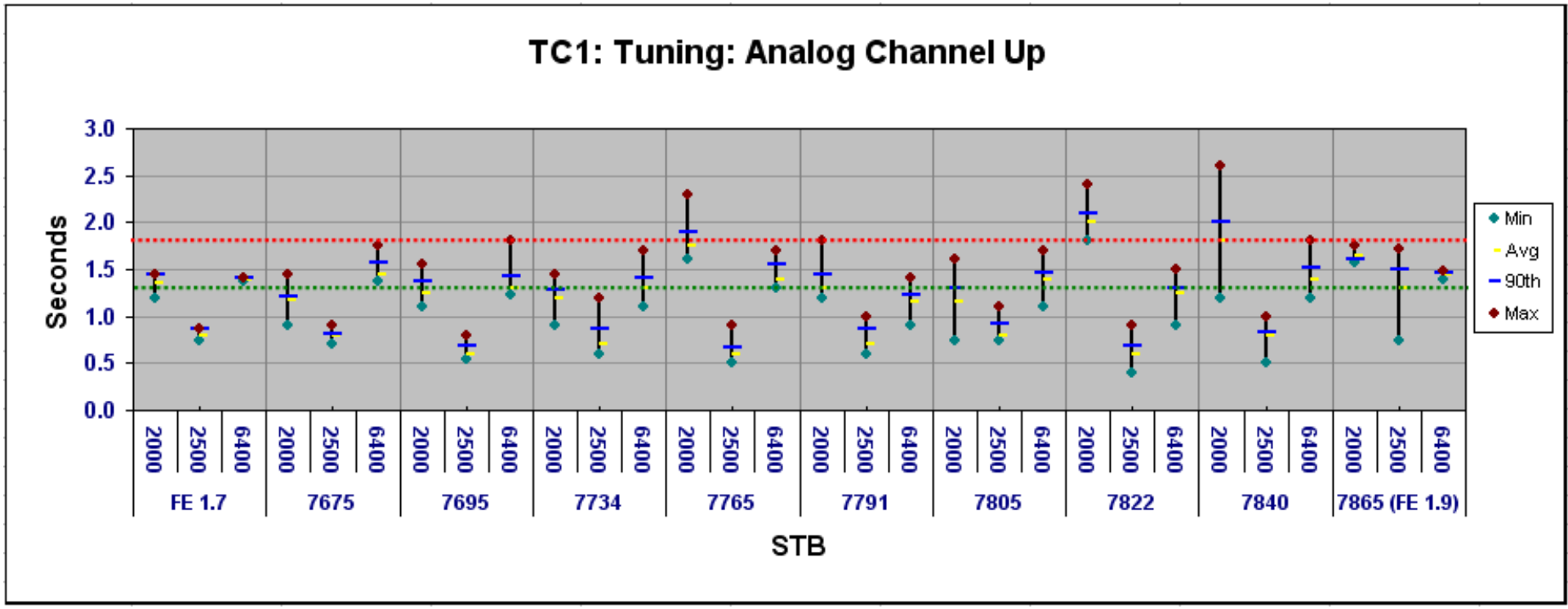
Power Corrupts...

PowerPoint Corrupts Absolutely.

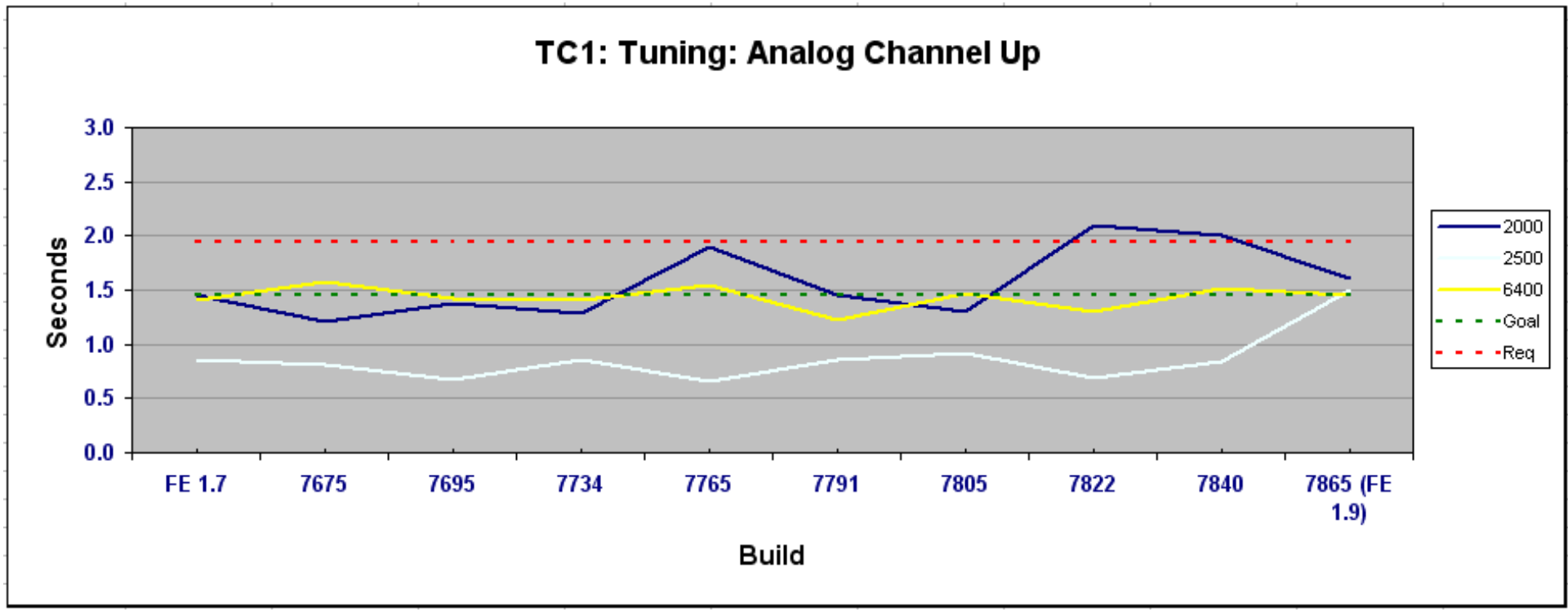
Relative Performance



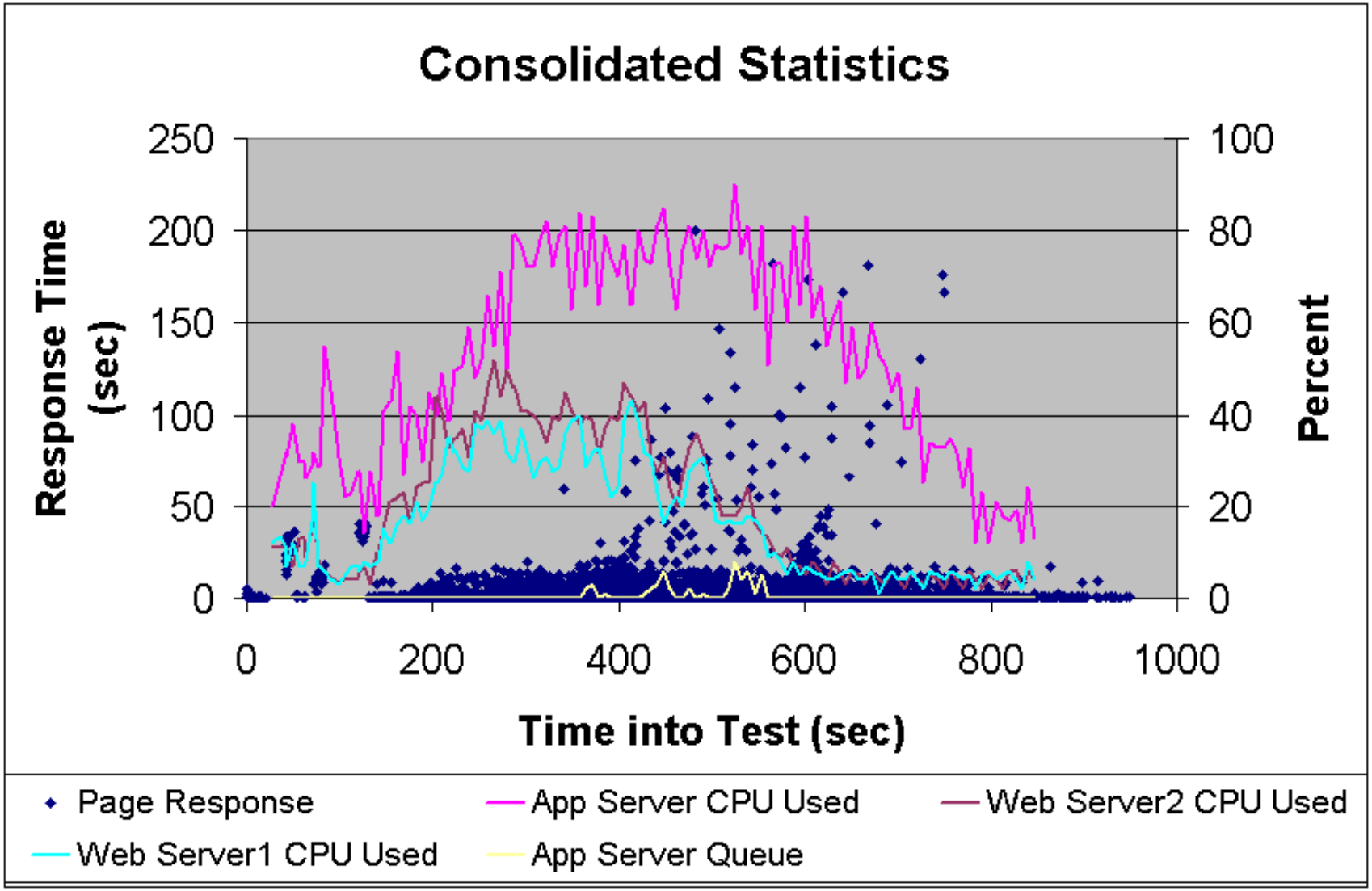
Graphs Make Some Things Obvious



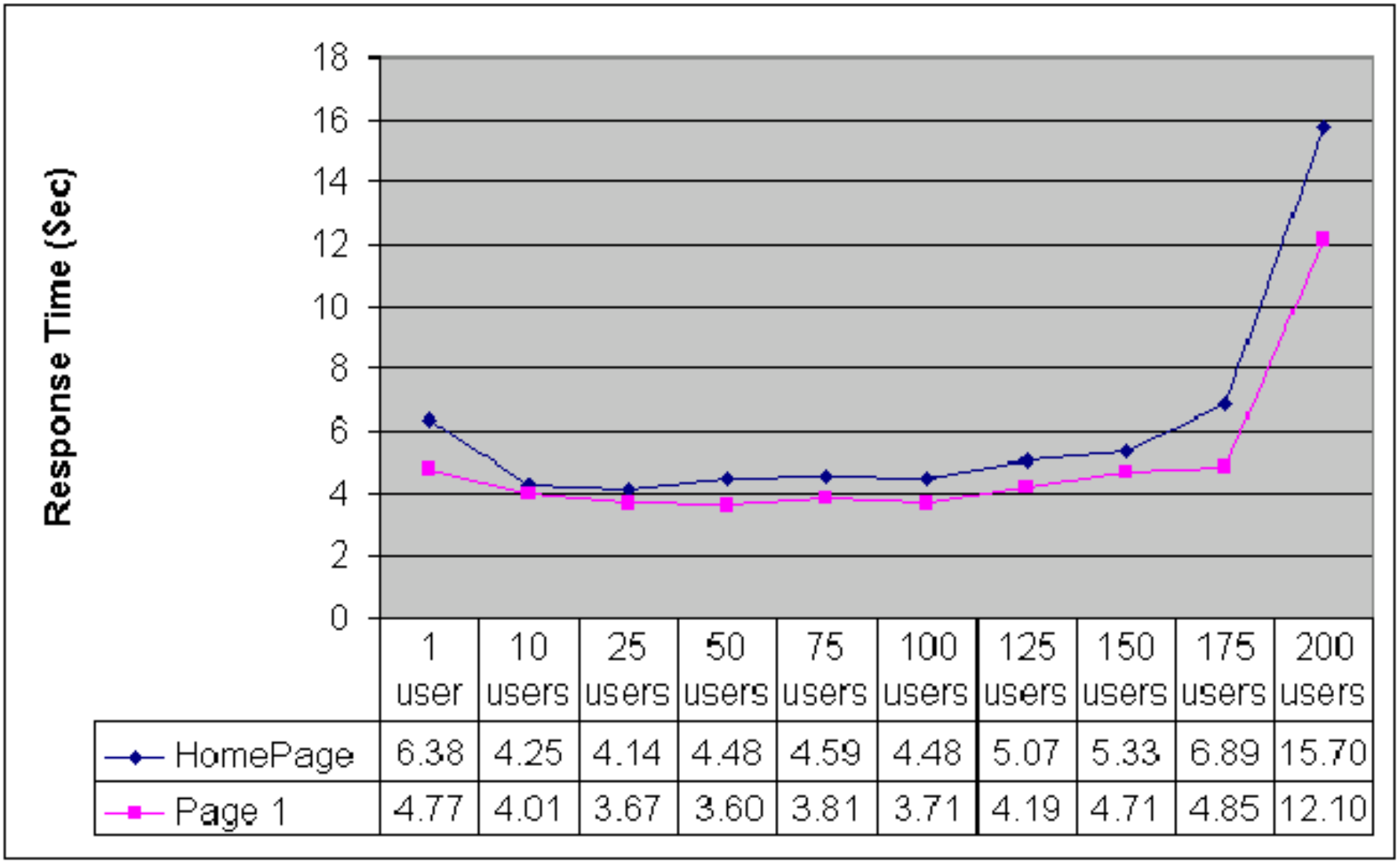
Trends, Trends, Trends!!!



Report

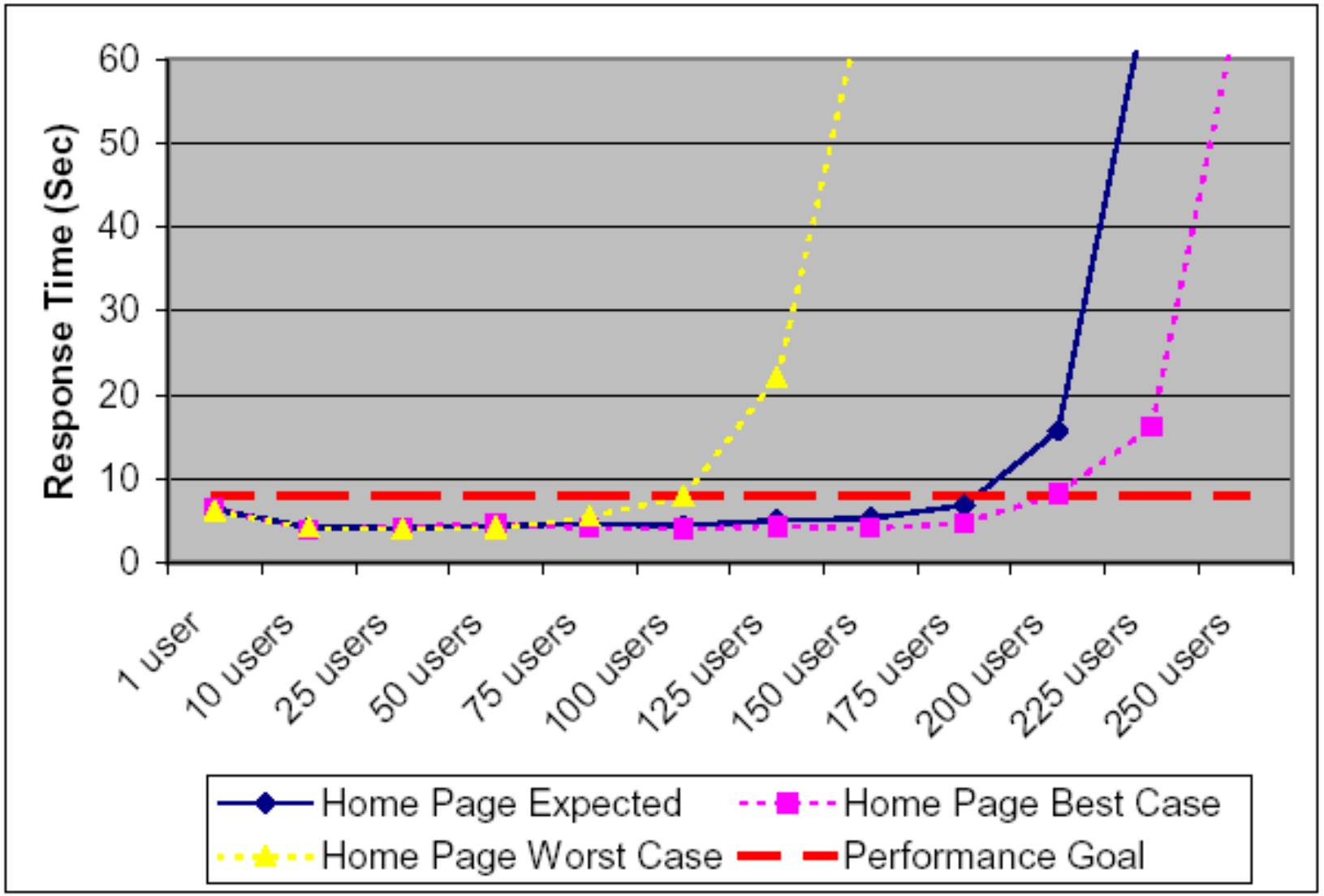


Report



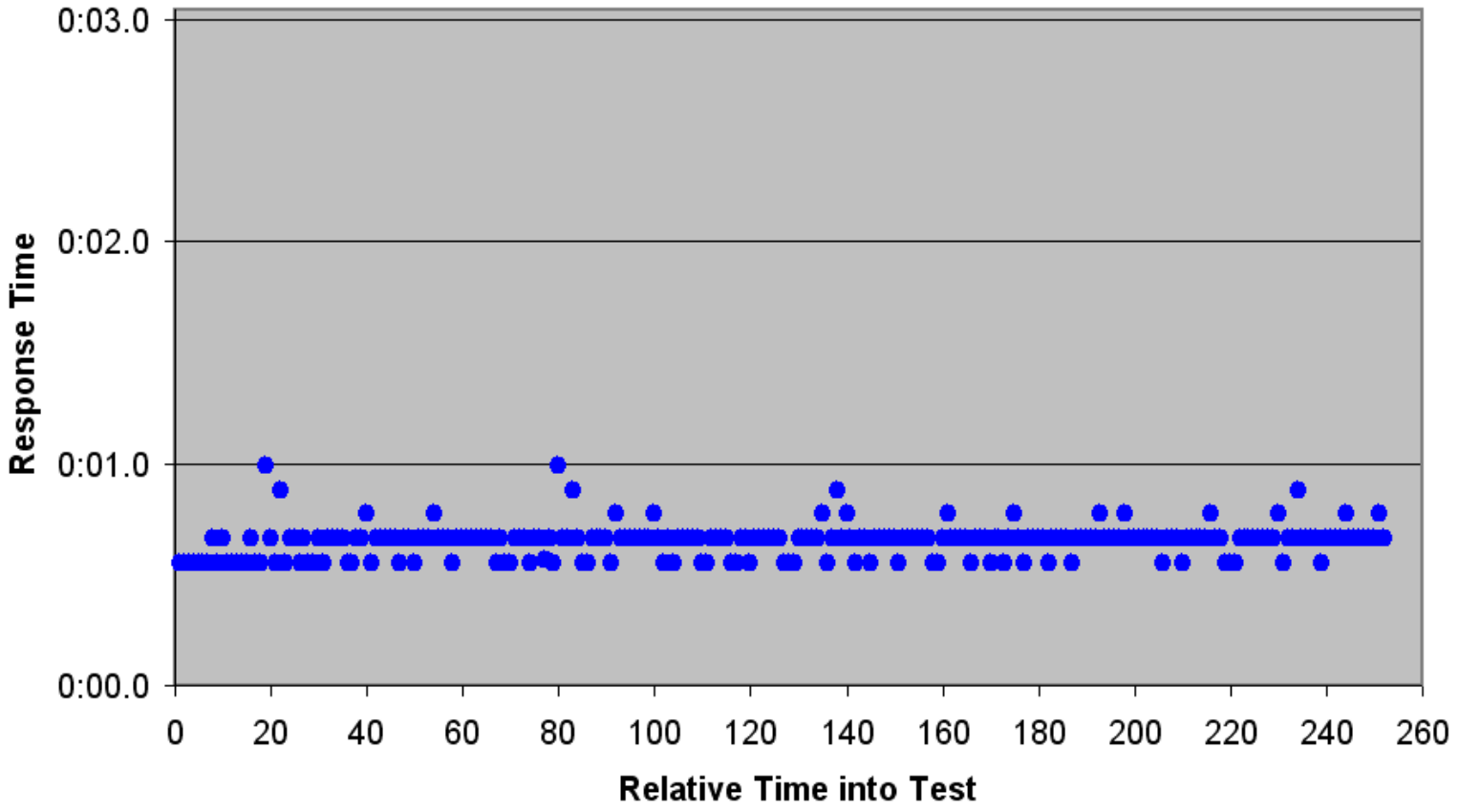
	1 user	10 users	25 users	50 users	75 users	100 users	125 users	150 users	175 users	200 users
HomePage	6.38	4.25	4.14	4.48	4.59	4.48	5.07	5.33	6.89	15.70
Page 1	4.77	4.01	3.67	3.60	3.81	3.71	4.19	4.71	4.85	12.10

Report

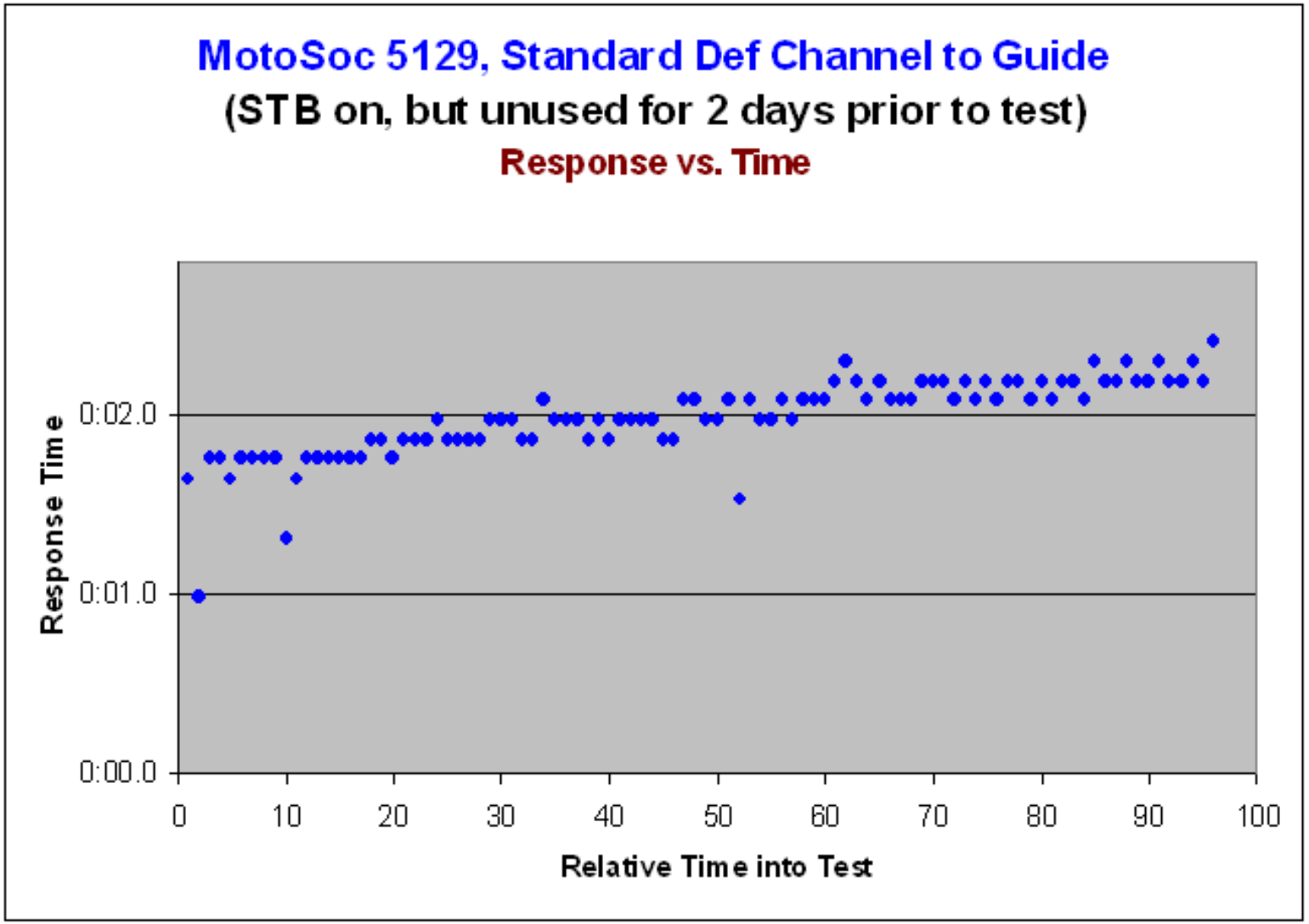


MotoSoc 5149, Standard Def Channel to Guide

Response vs. Time



	Seconds
Avg:	0:00.64
Median:	0:00.66
90th:	0:00.66
STD:	0:00.07
Min:	0:00.55
Max:	0:00.98

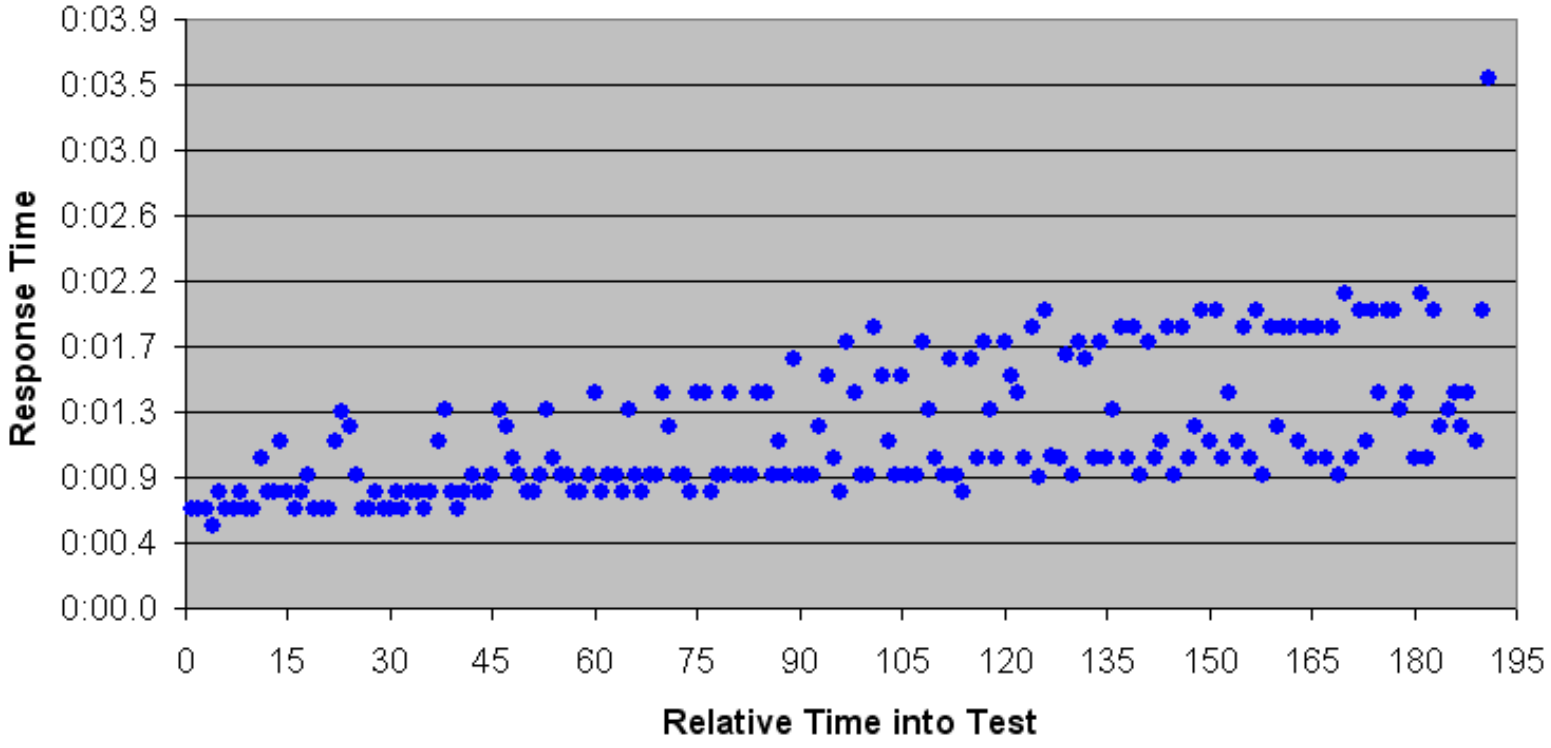


	Seconds
Avg:	0:01.98
Median:	0:01.97
90th:	0:02.19
STD:	0:00.22
Min:	0:00.98
Max:	0:02.41

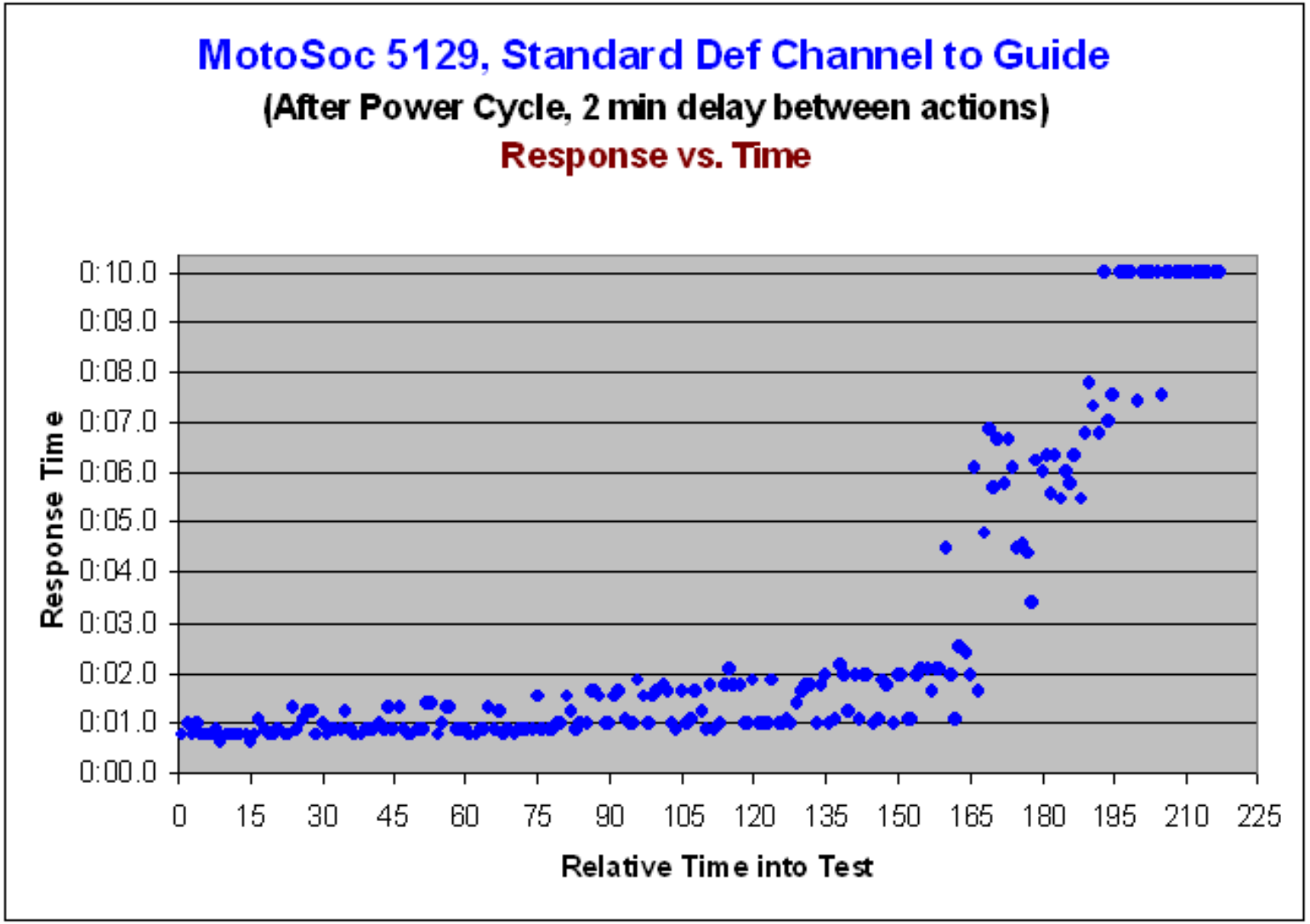
MotoSoc 5129, Standard Def Channel to Guide

(After Power Cycle, until Box became unresponsive)

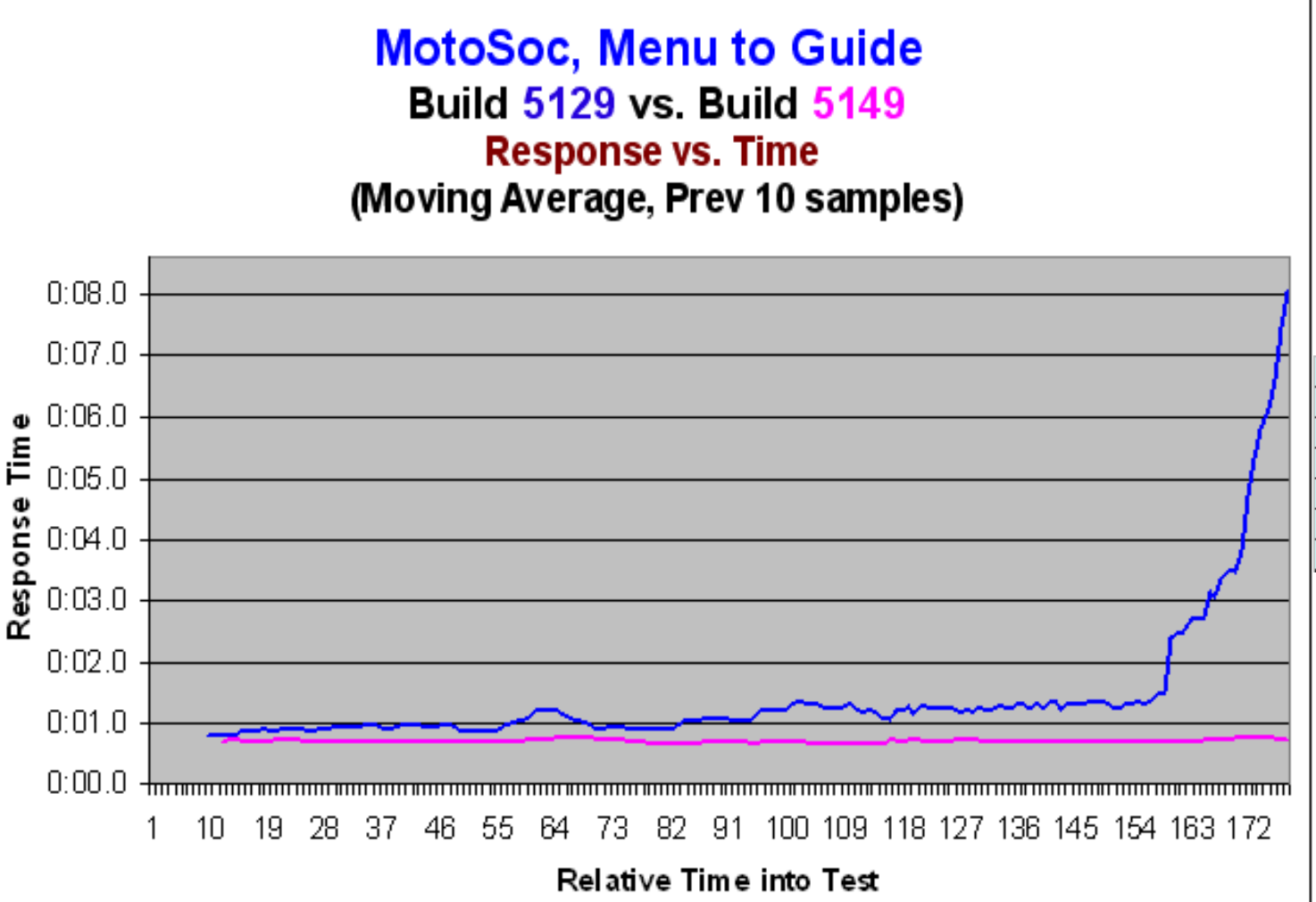
Response vs. Time



	Seconds
Avg:	0:01.16
Median:	0:00.98
90th:	0:01.86
STD:	0:00.45
Min:	0:00.55
Max:	0:03.50



	Seconds
Avg:	0:02.75
Median:	0:01.20
90th:	0:07.63
STD:	0:02.97
Min:	0:00.66
Max:	0:10.00



	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

Instructions:

Reassemble into your group.

Spend a few minutes discussing and sketching 1 graphic that would convey the key information to stakeholders that you don't believe they are getting now.

Be prepared to let the class assess your graphic.

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?"*

Performance 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.
Install	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.

Questions

Contact Info

Scott Barber

Chief Technologist

PerfTestPlus, Inc

E-mail:

sbarber@perftestplus.com

Web Site:

www.PerfTestPlus.com