PerfTestPlus
Better Testing... Better Results

Load Models for Performance Testing with Incomplete Empirical Data

Load Models

Abstract: For all that it is a common topic of conversation and concern among people and organizations that build or depend on web-based applications, testing a website in such a way that the test can reliably predict performance is still often more of an art than a science. More than a few brilliant minds have dedicated their careers to this complex topic. Several have published detailed and mathematically sound methods to plan for, predict and model performance characteristics very accurately… as long as there is sufficient empirical data to work from.

In this paper, we will first briefly explore the concepts, strengths and applications of Connie Smith's Software Performance Engineering (SPE) methods for creating well performing web-based applications. Then we'll look at Alberto Savoia's approach to creating representative load models from the analysis of existing patterns to then be applied by load generation tools. Next, we'll look at Daniel Menasce's techniques for building load models of existing sites that both predicts bottlenecks before they happen and generate the data necessary to prevent or push out that bottleneck before the real load gets great enough to show any symptoms of the bottleneck. Then, we'll look at how J.D. Meier ties these methods together into a single end-to-end approach. As you will see, these are all very powerful tools that performance testers and analysts have at our disposal.

After reviewing these methods, you will also see that all of these approaches depend on a fairly significant amount of empirical data that is rarely available to the individuals who are doing most of the performance testing and analysis of web-based applications in the industry today – consultants. That is where the insight in this paper actually begins, with the question of "How do I predict actual performance, when I don't have enough data to model the load accurately?" We will explore various methods that have proven valuable both individually and collectively for some of the top performance test/analysis consultants in the world. Finally, we will tie it all back together by demonstrating how parts and concepts from each of these methods can be blended together when there is not enough data to apply them exactly as described by their authors, based on the context of the task at hand, to *Create Effective Load Models for Performance Testing with Incomplete Empirical Data*.

by:

R. Scott Barber

SPE Overview:

In her book *Performance Solutions: A Practical Guide to Creating Responsible, Scalable Software*, Connie U. Smith, PhD., details how to apply the discipline known as Software Performance Engineering (SPE). She describes this discipline in this way:

> *"SPE is a comprehensive way of managing performance that includes principles for creating responsive software, performance patterns and antipatterns for performance-oriented design, techniques for eliciting performance objectives, techniques for gathering the data needed for evaluation , and guidelines for the types of evaluation to be performed at each stage of the development process.*
>
> *"SPE is model-based… By building and analyzing models of the proposed software, we can explore its characteristics to determine if it will meet its requirements before we actually commit to building it."*[1]

I often refer to SPE as an "Architect for Performance" method. This method includes:

- Modeling the performance of critical use cases by estimating performance characteristics such as:
  - Number of CPU cycles needed per activity
  - Amount of Disk I/O required
  - Number of messages sent between components or tiers
  - Time taken by each of these activities
- Summing the total estimated time and resources used by each activity

Devices	CPU	Disk	Display	Delay	Net
Quantity	1	1	1	1	1
Service Units	Sec.	Phys. I/O	Screens	Units	Msgs.
Screens	0.001		1		
Host	0.005			3	2
Log	0.001	1			
Delay				1	
Service Time	1	0.02	1	1	0.05

Figure 1: Example Overhead Matrix[2]

- Modeling the entire application (via UML)
- Grouping and adding the activity times and resources from the user's perspective
- Comparing those composite metrics to the performance goals for a single user
- Adding up those metrics across the expected number of users and usage patterns to evaluate if the proposed architecture will have the resources available to handle the anticipated volume.
- Validating initial estimates as software is being built, tuning software and hardware and modifying the model as needed.

[1] Smith, 2002
[2] Smith, 2002

I have seen these concepts applied in the field both extremely effectively and extremely poorly. All of the teams that applied SPE with superior results took to heart the single paragraph (section 15.3.3) in the book that addresses the core of this paper, which says:

> *"Performance testing is vital to confirm that the software's performance actually meets its performance actually meets performance objective, and that the models are representative of the software's behavior."*[3]

The reason this is so vital is because models are not flawless and without confirmation of the model's predictions one cannot know if there are flaws in either the model or in the implementation of that model. Naturally, this assumes that the performance test itself is accurate and while Chapter 3 details how to model the usage of the application in terms of performance with UML, the book does not discuss how to determine or estimate actual usage of the application under test.

Savoia Overview:

> *"In order to be worth anything, I believe that Web site load tests should reproduce the anticipated loads as accurately and realistically as possible. In order to do that you will need to study previous load patterns and design test scenarios that closely recreate them. This is a task that will require a serious amount of hard work, intelligence, intuition, and communication skills."*[4]

In his 2001 article for STQE, *Web Load Test Planning*, Alberto Savoia outlines his approach for determining the anticipated load of an application, and creating load tests to accurately represent this load. His preferred method of determining actual load is by analyzing log files from the existing version of the application. Unfortunately, in the field, most applications being tested either are first generation applications, or there are no log files to analyze.

In the absence of relevant log files, Savoia recommends generating log files via use of a limited beta release of the application to a representative sampling of users. Savoia goes on to demonstrate how to create what he terms a "**Web site Usage Signature**" or **WUS**. **WUS** is a method of extracting a model of the web sites actual usage based on user sessions and page request distributions. He goes on to discuss how to estimate overall traffic growth and peak usage levels. Each of these methods have been in common use by performance testers since **WUS** was first publicized and are commonly considered to be the most straightforward and accurate method to create performance test models where relevant usage data (in this case log files) exist and are accessible to performance testers.

Over the past several years, I have had sporadic communication with Alberto Savoia, during which time we have discussed this and his other performance testing theories. During these discussions, he agreed that without actual usage data, empirical data, his WUS will not help you determine actual usage. When asked how often he experienced this situation, he said 'very rarely'. My experience, of nearly 50 performance testing projects and indirect experience with over 100 more, I have only once had the

[3] Smith, 2002
[4] Savoia, 2001

opportunity to analyze log files from a similar enough previous version of the application under test for it to accurately predict its future use. In all other cases, additional data was required. During our discussions, we came to the conclusion that the reason for this difference was that his experience was filled with high-end clients, before the internet became a mainstream Business to Client tool while my experience is filled with more 'typical' clients of the dot-com era. His clients were willing and able to fulfill his request for data collection through beta-release, mine didn't have the money to conduct performance testing, let alone conduct an extended, unscheduled beta test and couldn't afford to delay release due to competitive pressures. The end of the dot-com era has not (at least not yet) increased budgets nor extended release schedules, once again resulting in not having enough empirical data, in the majority of cases, to apply this approach as written.

Modeling for Capacity Planning Overview:

Daniel Menasce is an ACM fellow and a university professor who is widely recognized for his academic contributions to application performance. His methods and approach have been repeatedly hailed as both detailed and accurate. In his book *Scaling for E-Business*, he takes a detailed look at capacity planning and scalability. The approach begins with models of both the application's usage and architecture. His usage models are similar to Savoia's **WUS** and his architectural models are similar to Smith's, but both rely on existing empirical data. Later, he mentions that forecasting future load levels and usage models is critical to capacity planning and details several mathematical methods to conduct that forecasting including Regression, Moving Averages and Exponential Smoothing. Of course, each of these methods requires a significant and accurate base of historical data.

Given actual usage data to evaluate current performance metrics, and a forecasting algorithm that yields high confidence predictions of future usage (and no significant changes to the application's architecture and/or functionality during the period of the forecast) Menasce's methods of applying queuing theory and performance laws to capacity planning are nearly infallible. Yet again, however, without empirical data these theories, laws and equations will only produce results and predictions as accurate as the input data.

Overview of an End-to-End Approach:

There are surprisingly few publications that discuss the integration of these phases of an application's performance lifecycle. Several publications discuss one of the phases in detail and mention that the other two are important, but don't actually demonstrate how it is accomplished. J.D. Meier of Microsoft has produced the first publicly available literature that includes detail about the application of all three phases to a "real-life" in the 1120 page MSDN guide titled *Improving .NET Application Performance and Scalability*.

This guide takes the reader through the entire life cycle of a .NET application, from conception through maintenance, with respect to performance. While not explicitly stated, much of the guide implies that expected load and usage model is a given, but in the chapter dedicated to building performance tests, it states the following:

> *"You can determine patterns and call frequency by using either the predicted usage of your application based on market analysis, or if your application is already in production, by analyzing server log files. Some important questions to help you determine the workload for your application include:*
>
> - *What are your key scenarios?...*
> - *What is the maximum expected number of users logged in to your application?...*
> - *What is the possible set of actions that a user can perform?...*
> - *What are the various user profiles for the application?...*
> - *What are the operations performed by an average user for each profile?...*
> - *What is the average think time between requests?...*
> - *What is the expected user profile mix?...*
> - *What is the duration for which the test needs to be executed?..."*[5]

Even though each question above has roughly a paragraph of explanation and example associated with it, the answer to "How do I collect that data?" is not addressed.

Commonality of These Approaches:

While there are many commonalities across these approaches, the one that we are interested in for the purposes of this paper are those concerning the creation of models that accurately represent actual usage. In each case above, we found that while the author acknowledged that empirical data is not always available, they presented their methods and approaches assuming the performance tester would be able to obtain or create an accurate model in one way or another. Each method allows for some degree of estimation or "educated guessing", but none present a method for doing so. The assumption appears to be that a significant amount of empirical data will be available.

The other commonality that is worthy of note here is that each author's methods require performance testers to have a stronger background in both mathematics and computer engineering than only a small handful do. This means that a significant proportion of those who are actually ***doing*** the performance testing simply don't have the skills necessary to apply some of the techniques recommended by the authors. As much of the software development industry has started viewing performance testing as a commodity service, it has fallen on the shoulders of the "expert consultants" to teach the employee tasked with performance testing the application as an "additional duty" how to do an adequate job – in two weeks or less.

[5] Meier, 2004

Why Empirical Data is Uncommon in Practice:

The problem with this assumption is that it simply isn't consistent with the state of the practice. The majority of the companies doing performance testing are not the Amazon's, e-bay's and yahoo's of the world, but rather they are the edoc's, Cyberserve's and Indus International's. Each of these three companies were clients of mine that hired me less than 6 weeks before "go live" day with brand new applications, no approach to ensuring application performance and no empirical usage data available. This situation is the norm, not the exception.

Instructing these companies to do a beta rollout of 10-20% of their projected customer base to determine usage patterns would accomplish nothing more than convince them that I was not the right consultant for the job and lead them to choose someone who claimed either that they could estimate the usage levels and patterns for them, or that the actual usage pattern doesn't really matter for the purposes of performance testing.

The reasons for this are simple. Small companies:

- Can't afford the costs of the detailed approaches we've discussed.
- Can't (or aren't willing) to dedicated the time required to "do it right".
- Don't have staff who understand either the performance testing or the mathematical principals involved with the detailed approaches.
- Won't performance test often enough to justify hiring or training a full time performance tester.
- Don't feel they need that degree of accuracy. They "just want to know…."
    - Will my site handle 500 users?
    - Is it fast enough?
    - How many users before it crashes?
    - When it fails, where will I need to add another server?
    - Did I purchase enough bandwidth from my ISP?

The belief is that those questions can be answered in a few weeks (at most) for a few thousand dollars. While it ***may*** be true that any one of these questions could be answered in just a few weeks (assuming all the necessary information is immediately available, the environment is available and configured correctly, the necessary tools are in place, etc.) The truth is that most of these companies aren't even sure what the value is of the question they ***are*** asking, let alone what question they ***should be*** asking.

Develop an Accurate Performance Model Without all of the Data

It should be obvious that even though those who have written some of the most read, respected and referenced material on performance testing have not written a tangible method for handling a lack of empirical data, the testers in the field have been doing exactly that for years. Through a combination of my personal experiences with performance testing projects, collaboration with some of the top performance testing practitioners in the world and the significant feedback from the readers of my articles, I submit that the following approach is compatible with the methods of all the authors mentioned in this paper and provides both a reasonable approximation of actual performance as well as addressing the

concern of "what if our load and usage assumptions are wrong?". The approach does not require a significant background in mathematics to apply and can be applied quickly.

This approach can be summarized as follows:

- When testing a web site already in production, you can extract the actual usage, load values, common and uncommon usage scenarios (user paths), user delay time between clicks or pages, session abandonment and input data variance to name a few directly from log files. Many web sites are instrumented web traffic monitoring software such as WebTrends or LiveStat that can often provide much of this data directly.
- Whenever testing a web site with a significant amount of new features/functionality, use interviews. By interviewing the individuals responsible for selling/marketing the new features, you will find out what features/functions are going to be expected, and therefore most likely used. By interviewing existing users, you can determine which of the new features/functions they believe they are most likely to use.
- When testing a pre-production web site already the best option is Savoia's, which is to roll out a (stable) beta version to a group of representative users, roughly 10-20% the size of the expected user base and analyze the log files from their usage of the site.
- In the absence of log files, or the time and resources for a detailed log analysis, industry "standard" and/or competitor's metrics and statistics can often provide a valuable starting point. Companies such as Nielsen//NetRatings, Keynote, or MediaMetrix publish this type of data publicly on their sites.

United States: Average Web Usage
Month of April 2004
Work Panel

Sessions/Visits Per Person	68
Domains Visited Per Person	105
PC Time Per Person	80:05:13
Duration of a Web Page Viewed	00:01:01
Active Digital Media Universe	51,139,748
Current Digital Media Universe Estimate	55,341,981

Figure 2: Example Data from NetRatings

Keynote Consumer 40 Internet Performance Index

	Week of May 23 - May 29, 2004		Previous Week	
	Index 30.95		Index 29.22	
Web Sites with Best Performance Averages	Altavista [226]	5.30	Altavista [225]	5.32
	CSFBDirect [82]	6.99	CSFBDirect [81]	7.09
	Ameritrade [226]	9.70	Ameritrade [225]	9.68
	Fidelity [126]	12.17	Fidelity [125]	12.32
	Ask Jeeves [73]	13.16	Ask Jeeves [72]	13.28
	E*Trade [13]	14.12	E*Trade [12]	14.26
	Charles Schwab [44]	17.64	Charles Schwab [43]	15.95
	Yahoo! [12]	19.50	Yahoo! [11]	19.36
	Waterhouse [101]	19.84	Waterhouse [100]	20.28
	Orbitz	21.08	Hotwire [2]	21.21
Worst Average	(anonymous)	97.97	(anonymous)	61.09

The performance numbers above are in seconds. The number in parentheses after a web-site name is the number of consecutive weeks that the site has appeared in the top 10.

Figure 3: Example Data from Keynote

Top 10 Largest Gaining Advertiser Web Sites*
Percentage Change – Sunday, Feb. 1, 2004 versus Average of Four Previous Sundays
Total U.S. - Home, Work and University Locations
Source: comScore Media Metrix

	% Change in U.S. Visitors: 2/1/04 vs. Avg. of Four Previous Sundays
Cialis.com	1868%
Apple iTunes	593%
HRBlock.com	258%
PepsiWorld.com	190%
Dodge.com	139%
Cadillac.com	94%
TheTruth.com	72%
Ford.com	19%
WarnerBros.com	8%
SonyPictures.com	6%

*Limited to sites with greater than 10,000 visitors on Super Bowl Sunday

Figure 4: Example Data from Media Metrix

These charts alone provide a wealth of information upon with to start building a load and usage model. Although these figures are not specifically from the web site under test they work quite well as first approximations.

- If you have no log files and don't feel that the metrics from the companies above are representative, you can run simple in-house experiments using employees, customers, clients, friends, or family members to determine, for example natural user paths and the page-viewing time differences between new and returning users. This method is also known as the "clipboard and stopwatch" method, for obvious reasons. I have found this to be a highly effective method of data collection for web sites that have never been live, as well as validation of data collected using other methods.
- As a last resort, you can use your intuition, or best guess, to estimate based on your own familiarity with the site.
- Create visual models and circulate them to both users and stakeholders to review/comment. Ensure the model is intuitive to both non-technical users, technical designers and everyone in between. Also ensure the model contains all of the supplementary data needed to create the actual test.

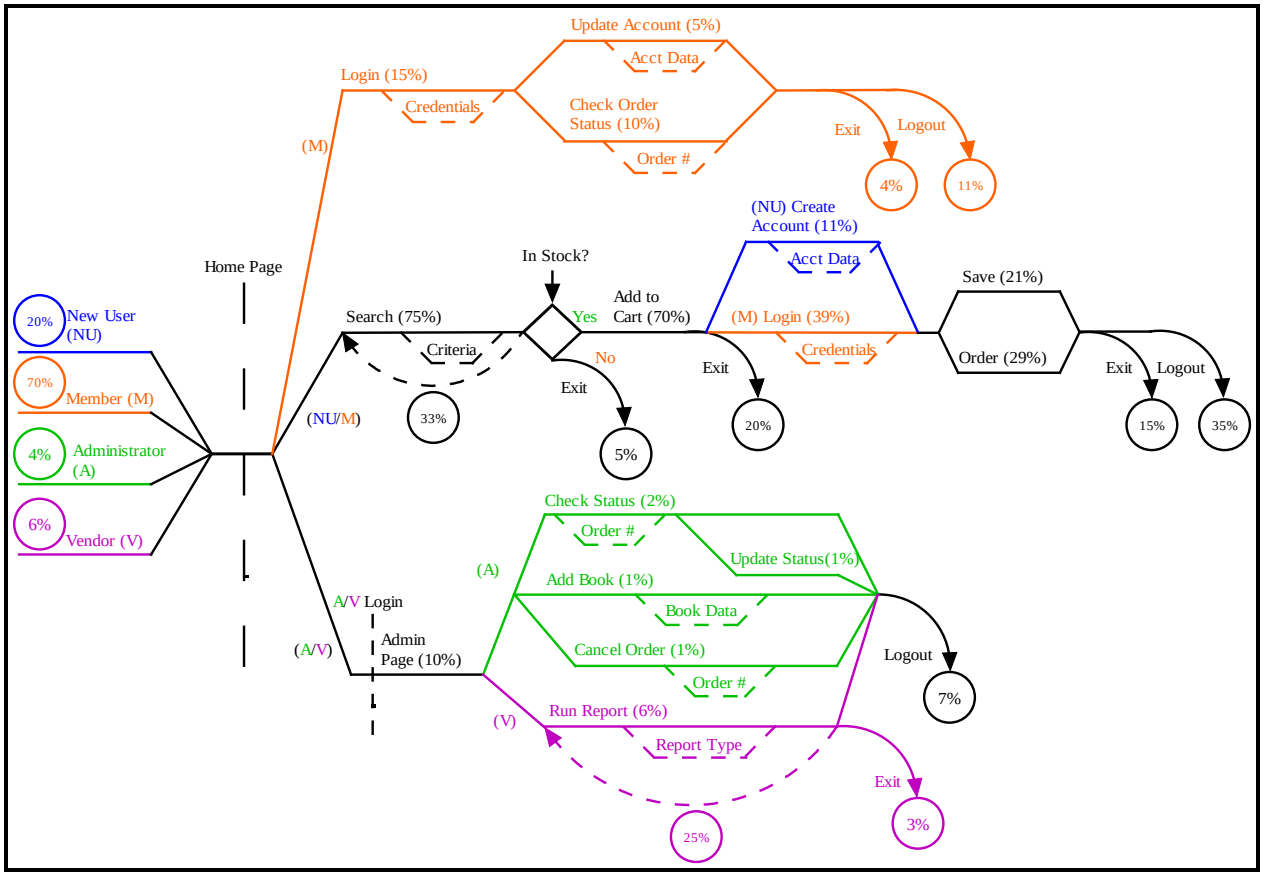

Figure 5: Example Visual Model for Online Bookstore

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

Figure 6: Supplementary Data for Online Bookstore Model

- Based on all data collected, create not fewer than three usage models (based on usage variance, not volume/load variance). Those usage models are:
  - Anticipated Usage
  - Best Case Usage in terms of performance (i.e. weighted heavily in favor of low performance cost activities)
  - Worst Case Usage in terms of performance (i.e. weighted heavily in favor of high performance cost activities)

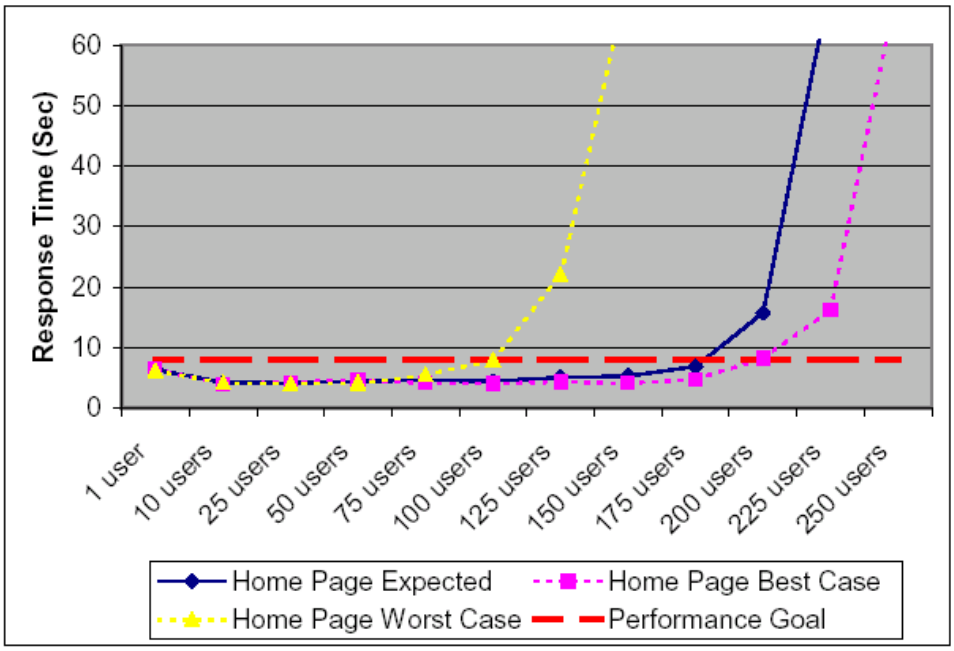

Figure 7: Sample "Best/Expected/Worse" Case Results Chart

- Model load patterns both separate from and together with usage patterns over time.

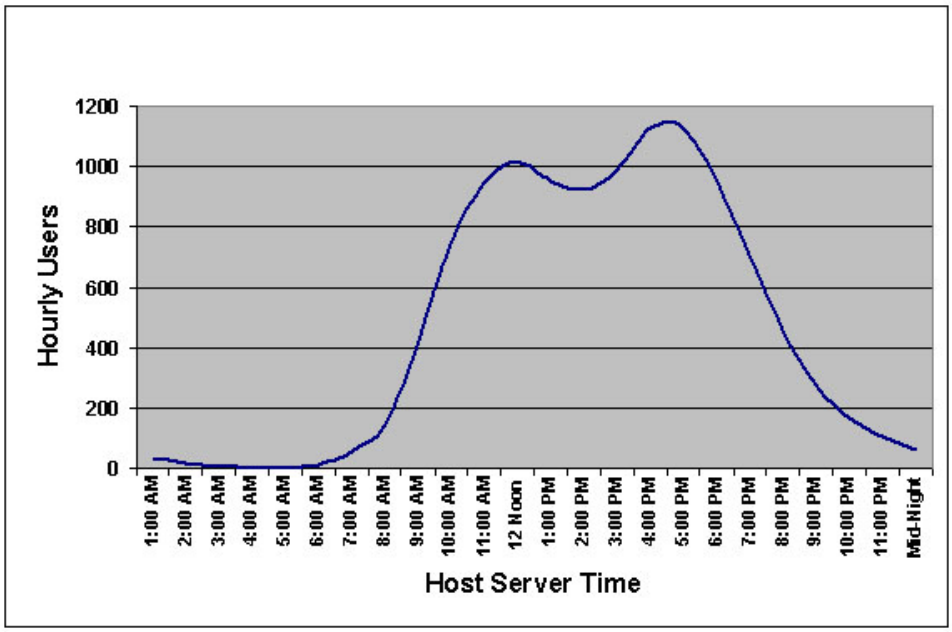

Figure 8: Sample Load Model over One Day

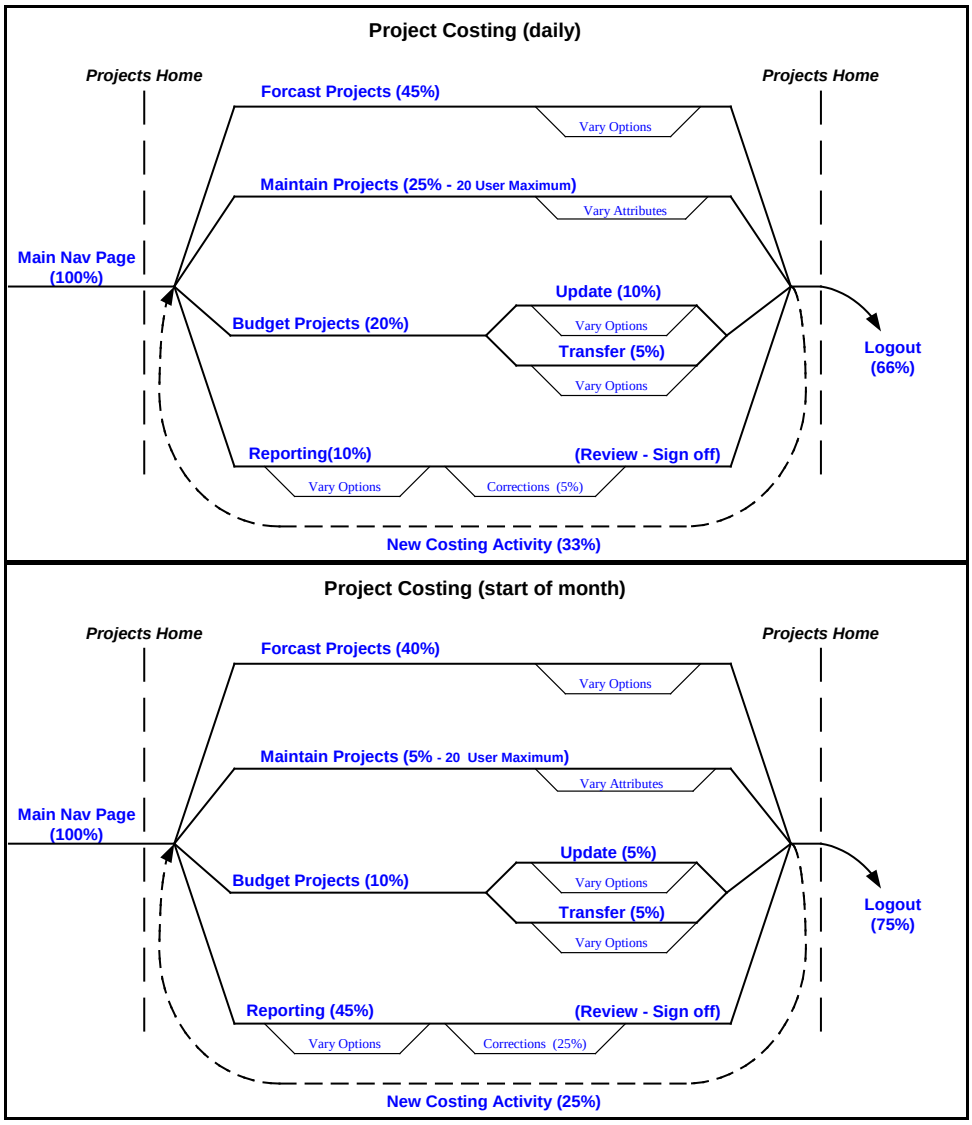

Figure 9: Sample Usage Model Variance by Time of Month

- Present results from each usage model to the stakeholders along with the models and be prepared to modify the model. Experience shows that by the time the results are gathered, the best available information about the anticipated web site usage has often changed.

An important note to this approach is that it's critical to model system-intensive activities even if they aren't the most popular activities on the web site. For instance, if the “managing business accounts” represents only 1% of all user activity, one could argue that it could be left out of the model. But what if this activity includes generating monthly usage reports that require the system to extract and compile large amounts of data from the database and then format it for display? While 1% of all users surfing to a static page won't cause a noticeable change in performance, 1% of all users running a monthly usage report may cause significant performance degradation. Therefore, this system-intensive activity must be included in at least one of the models.

Use with Methods Expecting Empirical Data:

(Treat data as if empirical, test with expected case until tuned to or near satisfaction, use as baseline, then test with worst/best case scenarios, compare)

Conclusion:

As critical as it is to creating load and usages models that will predict performance accurately, the data necessary to create these models is typically not directly available to the individuals actually conducting the testing, if it exists at all. Since all of the generally accepted methods and techniques for testing and predicting application performance depends on usage and load models, an alternative to empirical data is needed, but none of the most referenced material about performance testing gives the tester an approach to follow if empirical data is unavailable.

In this paper, we explored an approach to creating load and usage models that is a quick and effective substitute for empirical data. This approach has been applied to and enhanced as a result of well over a hundred performance testing projects conducted by at least ten different consultants that I am aware of. In none of those cases was the approach to model creation blamed for unreliable or inaccurate results when used in conjunction with Smith, Savoia or Menance's published methods and techniques.

About the Author

Scott Barber is a prominent thought-leader in the area of software system performance and testing software systems in general, Scott Barber, founder and Chief Technologist of PerfTestPlus, makes his living writing, speaking, consulting, and coaching with the goal of advancing the understanding and practice of software testing. Scott has contributing to four books (co-author, *Performance Testing Guidance for Web Applications*, Microsoft Press; 2007, contributing author *Beautiful Testing*, O'Reilly Media; 2009, contributing author *How to Reduce the Cost of Testing*, CRC Press; 2011, author *Web Load Testing for Dummies*, John Wiley & Sons, Inc.; 2011), composed over 100 articles and papers, delivered keynote addresses on five continents, served the testing community as the Executive Director of the Association for Software Testing, and co-founded the Workshop on Performance and Reliability.

Today, Scott is focused on applying and enhancing his thoughts on delivering world-class system performance in complex business and technical environments with a variety of clients and is actively building the foundation for his next project: driving the integration of testing commercial software systems with the core objectives of the businesses funding the creation of those systems.

When he's not "being a geek", as he says, Scott enjoys spending time with his partner Dawn, and his sons Nicholas and Taylor at home in central Florida and in other interesting places that his accumulated frequent flier miles enable them to explore.