Accounting for User Abandonment in Performance Tests

(This paper has been adapted from *"Accounting for User Abandonment"*, Part 4 of the made for Rational Developers Network article series *"Beyond Performance Testing"*. The remainder of the currently completed articles in this series are available at www.rational.net and www.perftestplus.com)

By Scott Barber

"You should simulate user abandonment as realistically as possible. If you don't, you'll be creating a type of load that will never occur in real life—and creating bottlenecks that might never happen with real users. At the same time, you will be ignoring one of the most important load testing results: the number of users that might abandon your Web site due to poor performance. In other words, your test might be quite useless."

-"Trade Secrets from a Web Testing Expert"
http://www.keynote.com/downloads/articles/tradesecrets.pdf Alberto Savoia, for STQE Magazine May/June 2001

In this paper, we will explore performance testing issues related to user abandonment and how to create realistic user abandonment models. User abandonment is an area that is not commonly discussed when developing and analyzing performance tests. The intent of this paper is to expand on concepts introduced in Alberto Savoia's article quoted above.

User Abandonment

Before reading Alberto Savoia 's "Trade Secrets from a Web Testing Expert" a few years back, I did not consider user abandonment in my testing. After reading his article, I expanded on what I read and applied it in my own testing. What you are about to read paraphrases and extends Savoia's ideas about user abandonment presented in his article. So, having said all that, I get to explain to you in my words what user abandonment is (which is challenging since, I just reread Savoia's article and liked his words just fine). User abandonment, most simply, is when a user gets frustrated with a site, usually due to performance, and discontinues using that site, either temporarily or permanently. As we are all aware, different users have different tolerance levels for performance. The reasons users abandon are based on some of the same factors that we consider when we talking about determining performance requirements. Let's start by discussing why users abandon sites, how to determine their tolerances, and how to build an accurate user abandonment model for a particular website.

User Psychology

Abandonment is all about user psychology. One could make a case that the other topics I address below, usage considerations and user expectations, are just subsets of the rather

broad topic of user psychology. When it comes to abandonment, user psychology is more than just "Why *do* we abandon sites?" That answer is simple. "Because we get tired of waiting." The answer to the follow up question, however, is not so simple: "Why do we get tired of waiting?" Why do *I* get tired of waiting and subsequently decide to abandon? Here are a few of my common reasons.

- The site is just painfully slow
- I lose interest while waiting for a page to download
- I get distracted during download
- I figure I can find it somewhere else faster
- I just plain get bored
- While waiting I checked my email and forgot to go back

While all of those things can be summarized as "the site is too slow", the quantification of "too slow" can vary depending on the particular context. For instance, after about 8-10 seconds I start getting bored and move to another browser window. 8-10 seconds is not painfully slow, but it is slow enough to risk me abandoning.

Unlike when talking about performance requirements, the kind of user psychology we are talking about here is extremely variable, not only between different users, but between different visits by the same user. A user's tolerance might change dramatically from session to session. While this makes it even harder to predict when an individual might abandon, it also means that abandonment is more likely to follow standard distribution models such as normal or negative exponential distributions. We will discuss these in more detail below.

Usage Considerations

Usage considerations are even more relevant in user abandonment than they are to determining performance requirements. In previous a previous article, I shared my experience with filing my federal income tax return on line. I mentioned that I was more tolerant of slowness because my expectations had been set properly. While expectation setting may help keep me from getting annoyed as quickly, I wasn't going to abandon that site no matter how slow it got. The thought of losing my return and having to start over, or having an incomplete return submitted that might lead to me being audited would have kept me from abandoning. In contrast, when I am catching up on the news I'll abandon one site and check another long before I'll wait on a slow site.

When thinking about usage considerations, we are really talking about how much longer than a users typical tolerance they are willing to wait before abandoning. Usage considerations are actually little more than factors that we apply to a persons 'typical' tolerance based on the perceived importance of the activity being performed on line. Importance is a vague term that may apply in cases such as:

- Real or perceived loss of money as a result of not completing the transaction.

- Concern that inaccurate information will be submitted if the transaction is not completed.
- Not having an alternative means to accomplish the task at hand.
- Having a real or perceived deadline to complete the transaction.

Below we will discuss how we will determine and apply this "importance factor" to our abandonment solution.

It is worth noting that not all sites will have a real abandonment model. Many web-based applications that my co-workers and I have tested in the past are exclusively for internal audiences that have no choice but to wait. Take, for example, my current client. They have a policy that all employees (and consultants) must enter the hours they worked that week between noon and 5pm every Friday – unless they are not working that day, in which case they are required to notify their managers of what hours they worked so the manager can enter them into the system. With roughly 3500 employees and consultants accessing this system over a 5 hour period on top of the "typical traffic", this gets very slow. Under other circumstances, I would abandon the site and try later or go somewhere else. In this case, I have no choice but to wait. In this case, there is no abandonment model. I think this is the exception rather than the rule, but a common enough exception to point it out.

User Expectations

I have already mentioned user expectations in this paper. Abandonment is unquestionably affected by user expectations. My experience submitting my tax return is a prime example. The difference here is that in most cases we don't get to manage user expectations; rather, their expectations drive (or at least should drive) our requirements. We probably don't have the ability to poll the potential users of the system with a questionnaire like:

- What speed connection do you use?
- How fast to you expect this Web site to be?
- How long will you wait before abandoning your session?

If polling a subset of potential users is possible, we should probably do it. But, if we can't, how do we account for user expectations? Our best bet is to evaluate what the potential users are using now for the same activity and assume that they will become frustrated if this solution is noticeably (let's say 20% or more) slower than what they are using now. This is not a hard science, but doing this will give us a place to start while deriving our abandonment model. Hopefully, this thought process was part of your requirements gathering process and contributed to your overall performance requirements. If you collected requirements well I submit that your users' expectations should be (statistically speaking) equivalent to the performance requirement associated to that page, meaning that if the requested page downloads within the time specified in the performance requirements that user will not abandon the site for performance reasons.

System Non-response

Ultimately, no matter how patient a user may be, sometimes a Web site simply fails to respond. While this doesn't technically count as user abandonment (since they didn't make the choice to abandon, the Web site technically abandoned them) it does provide the upper bound for how long a user could potentially wait before ceasing their web surfing session. I'm using non-response as a catch-all term to cover such issues as:

- Secure session time out (requiring users to either abandon or start over).
- Browser "page cannot be displayed" timeout errors.
- Temporary (or permanent) Internet connectivity interruptions on either end.

Most load generation tools assume this category is the only time abandonment occurs. As Savoia notes in his article, this is highly misleading when trying to predict production level performance accurately. That is not to discount this category, but rather to put it in perspective with the entire picture of user abandonment.

Determining Abandonment Parameters

We've talked about factors that contribute to our abandonment model. These factors can be summarized into the following abandonment parameters:

- Performance Requirement
- Importance Factor
- Abandonment Distribution
- Absolute Abandon Time

Putting these factors into the table below will help organize them so we can easily make decisions about our ultimate abandonment model.

Page Name	Performance Requirement	Importance Factor	Abandonment Distribution	Absolute Abandonment
Home Page				
Pay Bill				
Search Web				
Submit Taxes				
Field Population				

Table 1: Abandonment Model Parameters (unpopulated)

For our purposes, I have entered five pages that we will determine user abandonment parameters for. These pages do not model any site in particular, but are samples of page types that have different abandonment parameters. Table 2 shows the populated version

of this table. Take a moment to review the table and then we will discuss how the parameters were determined.

Page Name	Performance Requirement	Importance Factor	Abandonment Distribution	Absolute Abandonment
Home Page	5 sec	Low	Normal	120 sec
Pay Bill	5 sec	High	Uniform	900 sec
Search Web	8 sec	Low	Negexp	120 sec
Submit Taxes	15 sec	Very High	Inverse Negexp	900 sec
Validate Field	3 sec	Medium	Normal	120 sec

Table 2: Abandonment Model Parameters (populated)

The Performance Requirement column is straight forward – simply copy the performance requirement from wherever you have it documented. If you have multiple requirements based on user connection speed you will want to follow this process and ultimately create an abandonment model for each connection speed.

You will see that the Importance Factor column isn't very scientific. It is simply a common sense assessment of the user perceived importance of that particular activity being accomplished before they abandon. You can certainly poll users and stakeholders to obtain this information, but I wouldn't spend the time to get more scientific than the 4-tier rating system (low, medium, high and very high) used here.

In the Abandonment Distribution column you will see the standard distribution types that we discussed earlier. If you are not familiar with these distributions, refer to User Experience, not Metrics: Part 2 "*Modeling Individual User Delays*". In fact, either a Normal (bell curve) or Uniform (linear) distribution will be most accurate in the majority of cases. All of you who have ever taken a statistics or psychology course know that almost everything that real human beings do (over a large enough sample) can be represented by a bell curve. You may also recall that the key to an accurate bell curve is the standard deviation. We know two things about standard deviations when it comes to web usage 1) they are exceptionally large (statistically) in comparison to the range of values and 2) they are almost impossible to calculate accurately by non-mathematicians. What that means is that in most cases we actually end up with a very "flat" bell curve that, in effect, approaches a linear distribution. Statistics aside, if you don't have a strong reason to do otherwise, choose between either a Normal or Uniform distribution based on your best judgment.

The Negexp (logarithmic, or one-tailed) distribution is much less common. The two examples I use here might even be a bit artificial, but bear with me. As I have mentioned, I was willing to wait as long as it took to submit my taxes, but if the performance had gotten bad enough, eventually the system would have ceased responding and I would have effectively abandoned. While it is likely that I would actually have waited until the system timed out, I might have abandoned sooner if I believed the system was not responding. This situation is represented by a one-tailed

distribution where a few users may abandon in a short period of time, but most users "hang in there" as long as possible.

The other example I have used is field population. What I mean by field population is when you are presented with a form and you start entering data and, for example, you chose a value from a dropdown and a whole bunch of other values on that form are automatically populated for you. Usually you don't even expect this to happen, but as long as it happens quickly (and the values that appear are correct) you don't mind – or at least, I don't. However, if it doesn't happen quickly all you know is that your page is frozen and you can't enter data in the next field (or worse, you can enter data, but it gets erased when the screen finally does refresh). That situation will cause users to abandon the site faster than any other situation I can think of. That is why I chose to represent user abandonment of that activity with a logarithmic distribution that has most people abandoning quickly and only a few people "hanging in there", that we call an Inverse Negative Exponential distribution.

Finally, we get to the Absolute Abandonment column. This is the most scientific of the parameters. This is simply the time after which either your browser stops waiting for a response (in this case 120 seconds) or your secure session expires and you have to start your session over (in this case 900 seconds, or 15 minutes). To determine these numbers, simply ask the architect/developer responsible for the presentation tier (or web server) to provide you with the information.

Creating an Abandonment Model

Now we get to create the actual abandonment model based on those parameters. This is really pretty simple process. If we ignore the Importance Factor, we are actually already there. See Table 3.

Page Name	Abandonment Distribution	Abandonment Min Time	Absolute Abandonment
Home Page	Normal	5 sec	120 sec
Pay Bill	Uniform	5 sec	900 sec
Search Web	Negexp	8 sec	120 sec
Submit Taxes	Inverse Negexp	15 sec	900 sec
Validate Field	Normal	3 sec	120 sec

Table 3: Abandonment Model – Minus Importance Factor

As you will see below, when we discuss how to apply this model in load generation scripts this is all of the information we need in our abandonment model. The tricky part is applying the Importance Factor. I have tried to come up with a "magic formula" for applying the Importance Factor, but it always became overly cumbersome and not particularly universally applicable. If you've been following either of my series' you

know that I like to keep things as simple as possible so, instead of formulas, table 4 shows some general guidelines to applying the Importance Factor.

Importance Level	Min Time Factor	Absolute Time Factor
Low	1	~ 1/4
Medium	~ 1.5	~ 1/2
High	~ 2	~ 3/4
Very High	~ 4	1

Table 4: Importance Factor Application Guidelines

Allow me to stress, these are guidelines; ensure you apply a healthy dose of common sense when applying these factors and take the context of your particular situation into account. There are two notes I want to make about these guidelines. First, you will notice that the Min Time Factor for Low Importance and the Absolute Time Factor for Very High Importance are both 1. This is by definition in our model. If you find yourself wanting to change those factors, consider re-assessing your *parameters* rather than the Importance Factor. Second, you will notice that for small ranges, applying these factors blindly could result in the Min Time being larger than the Absolute (max) Time. If this happens, simply revise the Importance Factors and re-calculate until your Min is once again smaller than your Absolute.

Applying those factors exactly to our abandonment model we come up with the model shown in Table 5. Take a moment to review the chart and form your own opinions about the times as they are listed.

Page Name	Abandonment Distribution	Abandonment Min Time	Absolute Abandonment
Home Page	Normal	5 sec	30 sec
Pay Bill	Uniform	10 sec	450 sec
Search Web	Negexp	8 sec	30 sec
Submit Taxes	Inverse Negexp	30 sec	900 sec
Validate Field	Normal	5.5 sec	60 sec

Table 5: Abandonment Model – Exact Importance Factor

As I review Table 5, I am quite comfortable with those values… except two. I do not believe that anyone is going to wait 450 seconds (7.5 minutes) for confirmation of a bill payment. This is probably because the session keep-alive is configured for longer than it needs to be (15 minutes vs. maybe 5 minutes), but assuming that configuration is non-

negotiable, I am simply going to arbitrarily change the Absolute Abandonment value for Pay Bill to 240 sec. I also do not believe anyone is going to wait 60 seconds for some fields to dynamically populate, so I am going to change that value to 20 seconds.

Does that mean that I disagree with my own guidelines? I don't think so. I think it just means that I am taking a step back to look at my model in the context of the unique aspects of the system I am modeling. (I strongly recommend that everyone take the time to do that throughout their projects.)

Finally, Table 6 shows the model that we will apply to our example script.

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

Table 6: Abandonment Model – Modified for Context

Effects of Abandonment on Performance Testing

Before I show you how to adapt your scripts to abide by the model we just created, let's pause for a minute to consider the effects of user abandonment on performance testing. As important as I think abandonment is to include in performance testing, I don't think it should be done blindly. Consider the following.

Not accounting for Abandonment

So what happens if we just don't account for abandonment at all? Simple – the script will wait forever to receive the page or object it requested. When it eventually receives that object, it moves on to the next object like nothing ever happened. If the object is never received, your script never ends. I can think of no value this adds to the performance testing effort – unless you have some need to show a stakeholder "Under the conditions you specified, the average pageload time was roughly 2.5 hours." Unfortunately, we do occasionally have to do something like that to make a point; however, that is exactly what we are doing in a case like that – making a point by using ridiculous and meaningless numbers. That is not a performance test, nor does it get you any closer to delivering a quality, well performing application.

The previous example is, of course, a worst case. If you do not account for abandonment and virtually none of your pageload times exceed your requirements then your test was perfectly accurate (in terms of abandonment simulation)… by accident. Don't settle for

being correct by accident, take the extra few minutes to include abandonment in your performance tests and have the confidence that your results are honestly accurate as opposed to accidentally representative.

Mis-accounting for Abandonment

Mis-accounting for abandonment is what load generation tools do by default. They assume that all users "abandon" at a pre-determined, yet still continue on requesting the following page like nothing happened. Of course, you can change settings to improve that by changing the time limit, or having the virtual user actually exit; but that is still not context specific by page. If you were really motivated, you could change the parameters before every request based on how long you think a user would wait for that particular object, but that still won't account for the page as a whole. Regardless, improper accounting for abandonment can result in results even more misleading than not accounting for abandonment at all. To support this statement, let's consider a few examples and their side-effects.

1) "At 240 seconds stop trying to get this object, log a message and move on to the next object" – If you have objects taking over 240 seconds to load this causes situations where subsequent objects may need previous objects to have loaded successfully and may cause unexpected errors because the tool is now "forcing" the application to serve pages that a real user could not have realistically reached. This will also skew page and object load times, because you don't actually know how long the object would have taken to load – yet that 240 seconds is calculated as if the download is successful. Worst of all, the subsequent errors normally mask the initial cause making it appear as if the script is flawed. This is all not to mention the additional load be applied after the timeout (that a real user would not be applying) that may be skewing your results.

2) "Just log when people would have abandoned for analysis, don't actually exit the virtual user." – While this may be useful during early testing (discussed in more detail below), it paints a very inaccurate picture of the actual abandonment rate for a laundry list of reasons, including:
    a. Once a virtual user gets one page slow enough to abandon, they usually get more if not exited, resulting in statistics showing an artificially high abandonment rate.
    b. Allowing a virtual user to continue that would have abandoned, keeps the current load on the system rather than letting them stop using the system and reducing the total load for others, which is going to cause more virtual users to experience response times in the abandon range, once again resulting in statistics showing an artificially high abandonment rate.
    c. If the abandonment level response time was actually do to an error, subsequent page requests may also produce errors making the actual problem (one slow page) much more difficult to detect.

Do note that we are talking about grossly misrepresenting real abandonment. Mis-modeling the abandonment ranges by a few seconds is not going to cause this kind of problem. Your abandonment model needs to be reasonable, not perfect.

Correctly accounting for Abandonment

Correctly accounting for abandonment provides some extremely valuable information to all of the stakeholders of the system and allows you to collect data to evaluate abandonment requirements, but those are far from the only benefits. For instance, with correctly accounted for abandonment you can see how your system/application behaves as the load grows. One of the great things about web sites is that if the load gets too great for the system/application to handle, it slows down, then people abandon, thus decreasing the load until the system speeds back up to acceptable rates. Imagine what would happen if, once the site got slow, it stayed slow until someone "fixed the server". Luckily, abandonment relieves us of that situation, at least most of the time. Assuming that the site performs "well enough" with a "reasonable" load, performance is generally self-policing. Unfortunately, this is at a cost of some lost customers/users.

I mentioned a minute ago that just logging potential abandonment and not exiting the virtual user may be useful during early testing. This is true for several reasons. For example, let's say that you have an abandonment model that says all users will abandon if they encounter a page load time of 30 seconds, but your site (under development) is taking an average of 45 seconds to return a page, even at very low user loads. You will still want your scripts to run all the way through to gather information and create system logs to help track down the reason the times are so slow. In this situation, abandoning all of the virtual users when they hit the homepage gives you no information to help tune the system. Use your best judgment early in testing about whether to just log or actually exit users when they reach the abandonment response time, but always exit users when you are executing test intended to predict the experience of real users in production.

Interpreting results

There are a few things I wanted to point out about interpreting results, even though this paper won't go into much detail.

1) Check your abandonment rate before you evaluate your response times. If your abandonment rate for a particular page is less than about 5%, look for and handle outliers. If your abandonment rate for a particular page is more than about 5%, you probably have a problem worth researching further on that page.
2) Check your abandonment rate before drawing conclusions about load. Remember, every user that abandons is not applying load. Your response time statistics may look good, but if you have 25% abandonment, your load may be 25% lighter than you were expecting.
3) If your abandonment rate is over about 20% consider "turning off" the exit feature and re-executing the test to help gain information about what is causing the problem.

Summing It Up

In this paper, we have explored the premise of user abandonment and determined how to model it.

Alberto Savoia concluded his 2001 article with this:

"When you adopt concurrent users as a load testing input parameter and fail to account for user abandonment you run the risk of creating loads that are highly unrealistic and improbable. As a result, you may be confronted with bottlenecks that might never occur under real circumstances."

It is now 2003 and user abandonment is still a relatively unheard of topic. I hope this paper will help increase awareness and encourage people to avoid the mistake of not taking abandonment into account during their performance testing.

Acknowledgements

Special thanks to Alberto Savoia for taking the time to review the article this paper has been adapted from.

About the Author

Scott Barber is a senior consultant for Noblestar and a member of Noblestar's specialty testing lab, NobleLabs. NobleLabs provides a variety of services in such areas as performance engineering, security engineering, and embedded device interoperability testing. With a background in network architecture, systems design, database design and administration, programming, and management, Scott has become a recognized thought leader in the field of performance engineering. Before joining Noblestar, he was a company commander in the United States Army and a government contractor in the transportation industry. Scott is an active participant on the Rational Developers Network Forums, and is a moderator for the performance testing and Rational TestStudio related forums on QAForums.com. Scott has built a website that compliments this series that you may visit at http://www.perftestplus.com to find more detail on some topics and view slides from various presentations he has given recently. You may address questions/comments to him on either forum or contact him directly via e-mail.