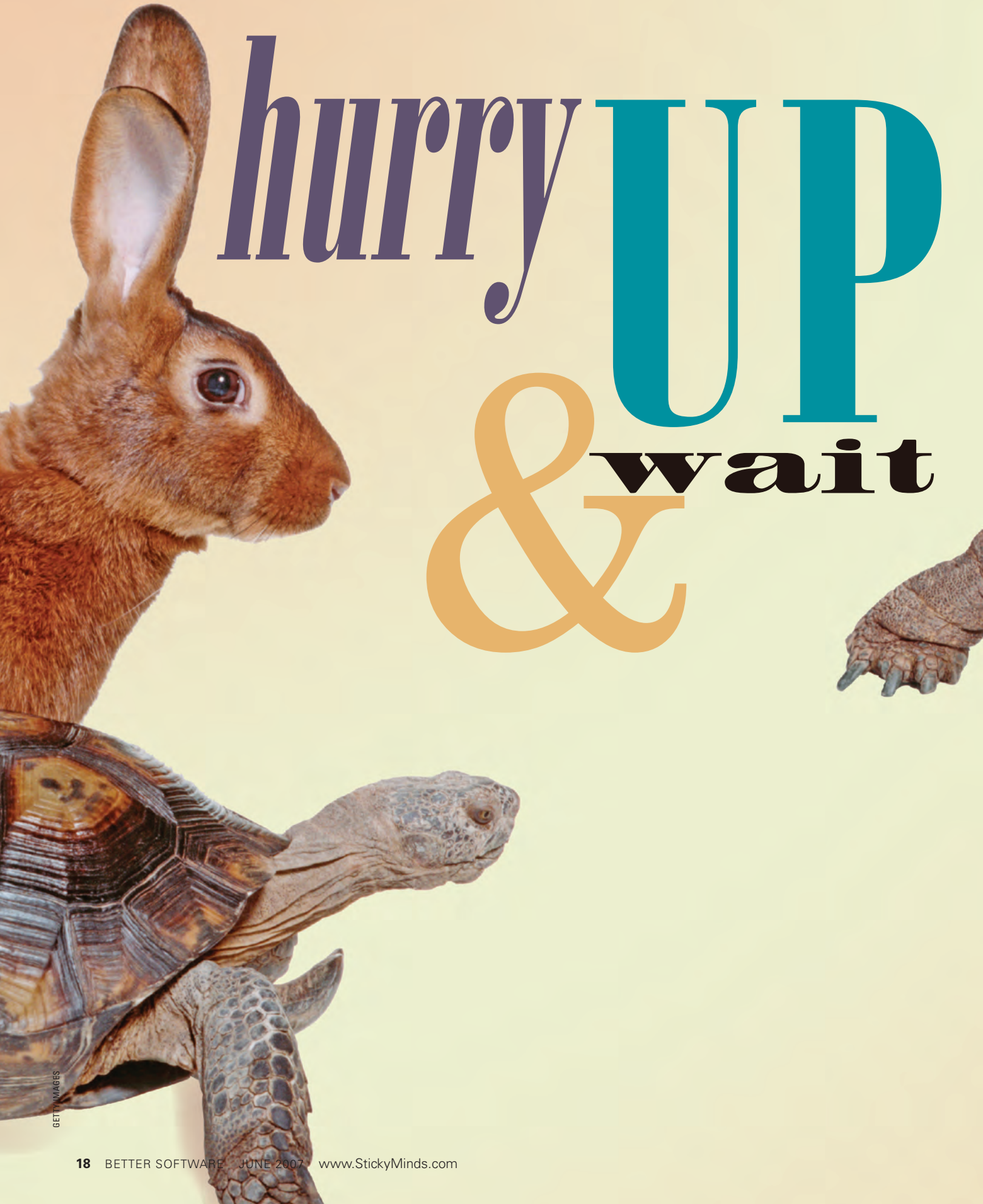
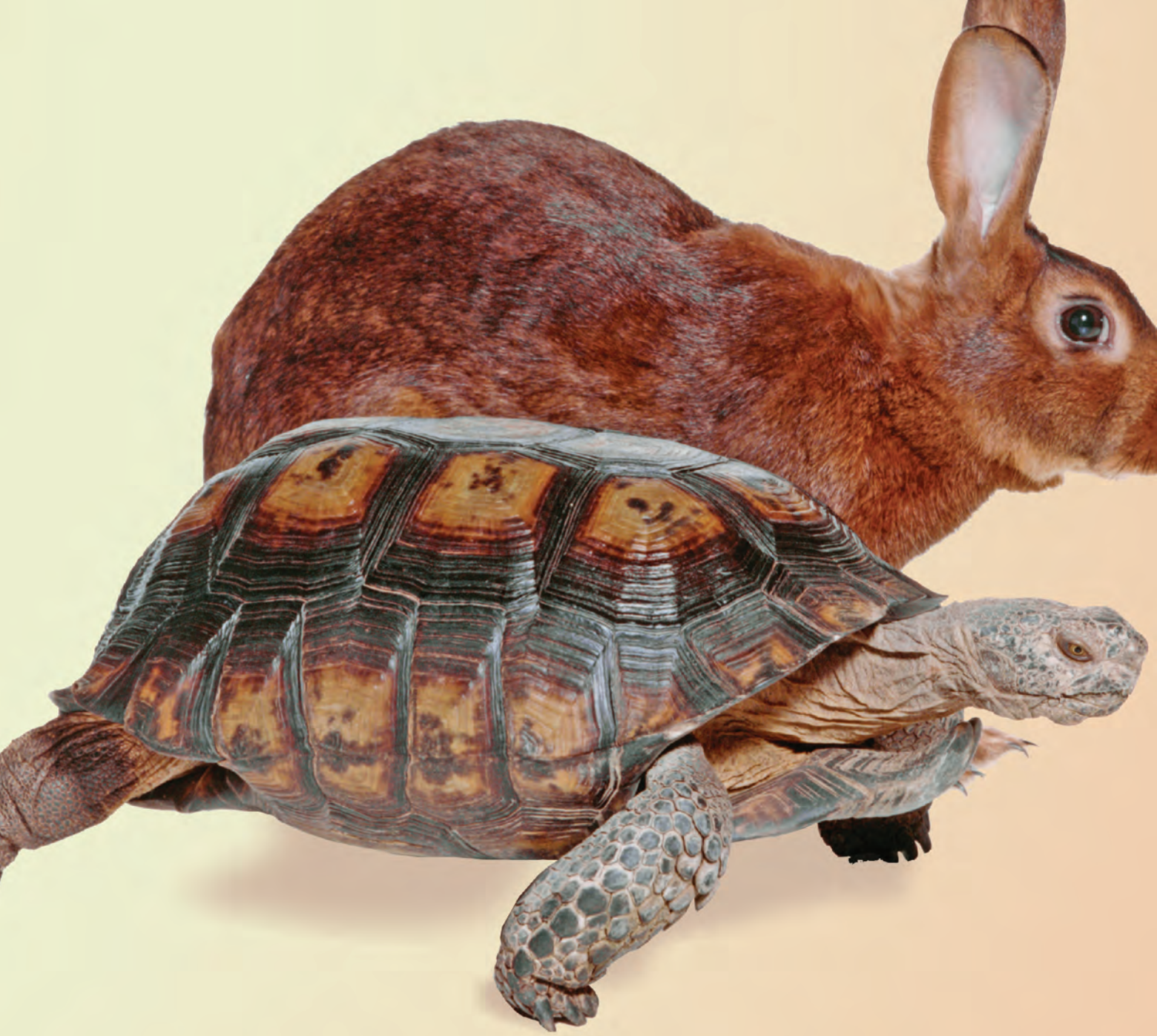
hurry UP & wait



GETTY IMAGES



When Industry Standards Don't Apply

By Scott Barber

No Industry Standard

It feels like hardly a single day has gone by in the past six years that someone hasn't asked me:

"What is the industry standard response time for a Web page?"

And in the past six years, the answer hasn't changed a bit. So, if the answer hasn't changed, why am I still getting asked the question on a daily basis?

The answer is simple: It's because there are no industry standards. How can there be? Think about how you use the Web. How long are you willing to wait for your homepage to display completely? How long are you willing to wait to view your family's online photo album? How long are you willing to wait for your tax software to confirm that your return has been submitted successfully? Are those numbers the same when you are at home as when you are at work? How about when you are using the wireless connection in an airport?

Your actual numbers don't really matter. The point is that no one number can possibly be the answer—at least until Web pages start regularly having response times that are under .25 seconds. Until then, what you are measuring is a combination of your current expectations about Web page response time and your determination to accomplish tasks via the Web.

You see, by the early 1980s, cognitive psychologists had determined that a delay of longer than one quarter of a second between an action and a response, whether on a computer or otherwise, would noticeably impact human performance, increasing error rate and the probability of a user's switching to a competing task. So as far as I'm concerned, until our Web sites make it to that .25 second barrier, the real question is not "What is the industry standard?" but rather "What response time will the users and stakeholders of my Web site or application find acceptable?" (See the StickyNotes for more on response time and user satisfaction.)

The Alternative to Standards

Experience reports from top performance testers shared during peer workshops such as the Workshop on Performance and Reliability (see the StickyNotes for a link) suggest that quantified performance criteria handed down from on high are sometimes met—but frequently ignored. These reports further suggest that even when performance cri-

on is not 'What is the

ut rather 'What response

d stakeholders of my Web

on find acceptable?"

teria are met, they rarely correlate to happy users if they can't relate back to qualitative criteria. Finally, these experiences suggest that determining the qualitative performance criteria enables teams to overcome quantification, technical, logistical, and managerial challenges related to achieving those criteria. This article explains how to determine and quantify those criteria. Before I do that, I'd like to make an important distinction between categories of performance criteria, namely requirements and goals.

Performance Requirements are those criteria that are absolutely non-negotiable due to contractual obligations, service level agreements, or critical business needs. Any performance criterion that will not lead unquestionably to a decision to delay a release pending proof of meeting that criterion is not absolutely required and, therefore, not a requirement.

Performance Goals are those criteria that are desired for release but have an inherent degree of flexibility. For instance, if a response time goal of three seconds is set for a particular transaction but the actual response time is determined to be 3.3 seconds, it is likely that the stakeholders will choose to release the application and defer tuning that transaction for a future release.

Before you can effectively determine the performance goals and requirements of an application, you will need to determine the transactions for which you are trying to characterize the performance. To be testable, goals and requirements need to correlate to some kind of user transaction. There are too many approaches to selecting transactions to address here, but the four most common and important categories are frequently used transactions, performance-intensive transactions, business-critical transactions, and transactions of special interest (possibly due to contractual obligations or stakeholder visibility).

Once you identify the transactions that are to have performance requirements and goals, it's time for you to engage the entire team—from executive sponsor to end-user—to determine exactly what those requirements and goals should be. You need to get their opinions, expressed in their own words, on how each transaction or group of transactions should perform. Once you collect that data subjectively, it becomes your job to convert that data into a testable form and record it as appropriate for your project.

Extract Requirements & Goals

There are several common sources of performance goals and requirements: project documentation and contracts, stakeholders, regulatory standards and competitive baselines, and users. Generally speaking, performance-requirement and goal-capturing efforts include the following activities (though not necessarily in any particular order):

1. Analyze written statements of requirements.
2. Determine whether or not competitive baselines are relevant to your project (capture them if they are).
3. Capture and analyze verbal statements of requirements (by people who have the power to make them stick).
4. Capture the opinions of other people internal to the company (opinions turn into goals).
5. Collect information from users (user goals).
6. Quantify the information gathered during activities 1 through 5.
7. Repeat the analysis throughout the project.

While it is generally desirable to capture requirements and goals early in the software development lifecycle, it is typically valuable to revisit those criteria

periodically throughout the project. No matter how well you capture performance criteria at the beginning of a project, contracts, perceptions, business drivers, and priorities change as new information becomes available. Additionally, you are likely to find that items initially classified as requirements become goals and vice versa. Keep that in mind as you traverse the project lifecycle. Finding out that the terms of a contract have changed while you are presenting what you believe is your final report puts your project in roughly the same place as never having known the terms of the contract in the first place.

WRITTEN STATEMENTS

As testers, we are used to analyzing documented requirements; the challenges in this case lie in obtaining and interpreting these documents. Often, contracts are viewed as proprietary and are not made available for review by non-executives. When facing roadblocks to reviewing contracts, it is important for you to be able to explain that the specific language and context of statements related to application performance are critical to determining compliance. For example, the difference between "transactions will" and "on average, transactions will" is tremendous. The first case implies that every transaction will comply every single time. The second case is completely ambiguous.

Frequently, the most important performance-related statements can be found in vision and marketing documents. Vision documents often hold subjective performance goals, such as "at least as fast as the previous release," "able to support a growing customer base," and "performance consistent with the market." Marketing documents are notorious for containing claims about product performance. I recommend treating any such claims as requirements because public statements can be used as ammunition by reviewers, users, or—in the worst case—lawyers hired by those users.

As I discussed at the beginning of this article, there are very few performance-related standards outside of life- or safety-critical devices and applications, but there are some. Remember to do your homework and find out whether or not any apply to your project.

Whether or not you are specifically requested to evaluate any of these items, stakeholders are almost always grateful when a performance tester validates—or invalidates—contractual obligations, marketing claims, and regulatory standards before the product goes live. In the same way, stakeholders typically want to know how their product compares to the competition, even if they don't specifically ask. Since I view providing stakeholders with data that will help them make informed decisions about the quality of the product to be a significant part of my job as a tester, I believe that testing against all of these items is important. Your role or mission may be different.

COMPETITIVE BASELINES

More frequently than having explicit or regulatory standards for performance, market expectations and competition create de facto standards. Every application in every industry vertical will have different methods and sources for determining its competitive landscape. For example, if your project is an eCommerce application, there are a large number of Web sites against which users will compare—either consciously or subconsciously—the performance of your application. In this case, it is likely valuable to choose one or more of the most popular eCommerce sites to compare your application against. Naturally, you can't run load tests against a competitor, but you can collect response time information by periodically surfing the site and from sites such as www.keynote.com. The bottom line is simply this: Don't assume your site won't be compared against others simply because there is no published standard; de facto standards are more likely to set your users' expectations than formal standards, anyway.

VERBAL STATEMENTS

As you know, stakeholders always have opinions when it comes to performance, and frequently they express these opinions in terms that appear to be quantified and absolute, even though they are rarely well understood. The key to collecting opinions from influential stakeholders is not only to capture the statements but also to determine the intent behind those statements. For example, a stakeholder with a background in telecommunications may say that she expects the application to have "five 9s of reliability." What she probably doesn't realize is that this equates to a near impossible standard for a Web site of being unavailable for roughly five minutes per year (about one second per day). The reality is that many Web sites could be down for an hour per day—if it's the "right" hour—without customers even noticing. In fact, it's hard to imagine that Web users would notice an additional one-second delay, even if it happened

once a day. It is easy to imagine a person being outraged by a dropped phone call at the same time every day due to a one second outage, but that same person wouldn't even notice the interruption in a Web connection. Asking good questions is the key to determining the real intent behind stakeholders' performance-related statements. In this particular case, the intent behind "five 9s of reliability" is uninterrupted service. So a more appropriate requirement may be "No more than 1 percent of users abandon their task as a result of excessively slow response times."

The process behind capturing the opinions of team members who cannot individually make their opinions stick is the same. The difference is that these opinions should be treated as goals rather than requirements. The reason for gathering these goals is that these people often have different perspectives of what is and is not acceptable performance. For instance, a customer service representative may tell you that the top priority is that customer information comes up quickly when he is assisting an upset customer. This statement could lead to an entirely different set of performance criteria for the portion of the application that the customer service representatives use than for the self-service portion that the customers use.

See the StickyNotes for sample starting questions and potential follow-up questions to help you capture the intent behind verbally expressed opinions.

SEPARATE REQUIREMENTS FROM GOALS

If you haven't done so already, it is useful to distinguish between requirements and goals prior to quantifying them. Requirements need to be much more carefully and completely quantified than do goals. A statement of "approximately three seconds to render the requested Web page," for example, is a perfectly respectable performance goal but a completely untestable performance requirement due to the ambiguity of the word "approximately."

Fortunately, identifying requirements is easy: Focus on contracts, legally binding agreements or standards, and any performance-related criteria that will cause the executive stakeholders to hold off on releasing the software until those criteria are met. These criteria may or may not be related to specific business transactions, but if they are, ensure that those transactions are included during performance testing.

At first blush, identifying goals seems purely mechanical; if it hasn't already been tagged as a requirement, it must be a goal. Your challenge is dealing with goals that contradict requirements. In this case, your best bet is to take these conflicting items to stakeholders for additional clarification. It may be that the goal is superseded by the requirement—in which case, you can simply remove the goal from the list. It may also be the case that the stakeholders will determine that the requirement is overly aggressive and needs to be modified. Either way, the sooner these conflicts are resolved, the less confusion they will cause later.

QUANTIFY GOALS

Some goals are relatively easy to quantify. For example, you can quantify a goal of "no slower than the previous release" by either referencing the most recent production performance monitoring report or by executing some single-user, light-load, and heavy-load tests against the previous release, recording the results to use as a baseline for comparison.

Frequently, the captured goals that need to be quantified aren't strictly comparative goals—they are user satisfaction goals. Quantifying end-user satisfaction or frustration is more challenging, but far from impossible. All you really need is an application and some representative users. You don't need a completed application; a prototype or demo will do for a first pass at quantification. For example, with just a few lines of code in the HTML of a demo or prototype, you can control how long it takes each page, graphic, control, or list to load.

Using this method, you can create several versions of the application with different response characteristics. Then have the users try each version, telling you in their own terms whether they find that version to be unacceptable, slow, reasonable, fast, etc., using whatever terms the existing goals use. Since you know the actual response times, you can start equating those numbers to the users' reported degrees of satisfaction. It's not an exact science, but it turns out to be good for goals—especially if you follow up by asking the same questions about satisfaction with the application every time you put it in front of someone. This allows you to collect more data and enhance your performance goals as the application evolves.

If your anticipated users are regular Internet users and you have designed your study to account for their typical connection speed, you probably will notice a tight grouping in your data. If, however, your anticipated users are not regular Internet users, their responses are likely to vary widely. In this case, you probably will want to analyze the data from those groups separately; however, focus on the regular Internet users because once people start using the Internet their expectations get reset very quickly.

The best thing about this method is that your team makes a great "control group" for a sanity check. Let's face it: Who complains more about a poorly performing Web site than people who design, develop, and test Web sites for a living? Typically, you will find that your team has slightly less tolerance for poor performance than your representative users, but the results should be fairly close. When the users are demanding better performance than your team or when your team is satisfied with dramatically worse performance than your users, you know your study—or your team's expectations about users—is flawed.

QUANTIFY REQUIREMENTS

If you are lucky, most of the performance requirements that you captured are already quantified and testable. If you are a little less lucky, the requirements that you captured aren't quantified at all, in which case you can follow the process described above for quantifying performance goals. If, however, you are unlucky, the performance requirements that you collected are only partly quantified and partly or wholly untestable.

If a requirement is extracted from a contract or existing marketing document, it likely cannot be changed. So when you are faced with a requirement such as

"three-second average response time," or "2,500 concurrent users," you have to figure out both what those requirements mean and what additional information is required to make them testable. There is no absolute formula for this. The basic thought process is to interpret the requirements precisely as written in common language, supplement them with the most common or expected state for the application, and then get your extended, testable requirement approved by the stakeholder who would be held responsible if someone were to legally challenge compliance with the requirements after the product goes live.

See the StickyNotes for examples of quantifying performance requirements.

REPEAT THE ANALYSIS

Though we may wish it were otherwise, performance requirements and goals are bound to change during the course of the project. Sometimes they get more stringent as more information becomes available about the needs of the customer, and sometimes they become less stringent as the pressure to beat a competitor to market with new functionality takes precedence over being faster upon release. This is normal and healthy (at least to a point; like most anything else, it can go too far). While it often is not necessary or valuable to redo the entire analysis from scratch, it is both important and valuable to periodically check back in with the people and documents that led to the goals and requirements in the first place to see what, if anything, has changed. As the product is nearing completion, it is often useful to bring the current performance measurements with you when checking in so that people can react to them. That reaction is often the best prerelease indication you will get of an individual's feelings about the performance of an application.

Final Thoughts

Until Web sites start displaying response times under .25 seconds, we need to conduct performance tests against legally binding requirements and make our best attempt to quantify the satisfaction of our users, remembering that users will be more satisfied with consistent performance than intermittently fast performance. **{end}**

Scott Barber is the chief technologist, CEO, and president of PerfTestPlus, Inc. His specialty is context-driven performance testing and analysis for distributed multi-user systems. Contact him at sbarber@perftestplus.com.

Sticky Notes

For more on the following topics go to www.StickyMinds.com/bettersoftware.

- Response time and user satisfaction
- Workshop on Performance and Reliability
- Sample questions
- Quantifying performance requirements