NN/g Nielsen Norman Group

Evidence-Based User Experience Research, Training, and Consulting

Jakob Nielsen's Alertbox: January 1, 1993

Response Times: The 3 Important Limits

Summary: There are 3 main time limits (which are determined by human perceptual abilities) to keep in mind when optimizing web and application performance.

Excerpt from Chapter 5 in my book Usability Engineering, from 1993

The basic advice regarding response times has been about the same for thirty years [Miller 1968; Card et al. 1991]:

- **0.1 second** is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except to display the result.

- **1.0 second** is about the limit for the user's flow of thought to stay uninterrupted, even though the user will notice the delay. Normally, no special feedback is necessary during delays of more than 0.1 but less than 1.0 second, but the user does lose the feeling of operating directly on the data.

- **10 seconds** is about the limit for keeping the user's attention focused on the dialogue. For longer delays, users will want to perform other tasks while waiting for the computer to finish, so they should be given feedback indicating when the computer expects to be done. Feedback during the delay is especially important if the response time is likely to be highly variable, since users will then not know what to expect.

Normally, response times should be as fast as possible, but it is also possible for the computer to react so fast that the user cannot keep up with the feedback. For example, a scrolling list may move so fast that the user cannot stop it in time for the desired element to remain within the available window. The fact that computers can be too fast indicates the need for user-interface changes, like animations, to be timed according to a real-time clock rather than being timed as an indirect effect of the computer's execution speed: Even if a faster model computer is substituted, the user interface should stay usable.

In cases where the computer cannot provide fairly immediate response, continuous feedback should be provided to the user in form of a percent-done indicator [Myers 1985]. As a rule of thumb, percent-done progress indicators should be used for operations taking more than about 10 seconds. Progress indicators have three main advantages: They reassure the user that the system has not crashed but is working on his or her problem; they indicate approximately how long the user can be expected to wait, thus allowing the user to do other activities during long waits; and they finally provide something for the user to look at, thus making the wait less painful. This latter advantage should not be underestimated and is one reason for recommending a graphic progress bar instead of just stating the expected remaining time in numbers.

For operations where it is unknown in advance how much work has to be done, it may not be possible to use a percent-done indicator, but it is still possible to provide running progress feedback in terms of the absolute amount of work done. For example, a system searching an unknown number of remote databases could print the name of each database as it is processed. If this is not possible either, a last resort would be to use a less specific progress indicator in the form of a spinning ball, a busy bee flying over the screen, dots printed on a status line, or any such mechanism that at least indicates that the system is working, even if it does not indicate what it is doing. Note added for the Web version of this essay: Most Web browsers fail in providing useful progress bars, since they don't communicate what percentage of the *entire download* for a page has been completed.

For reasonably fast operations, taking between 2 and 10 seconds, a true percent-done indicator may be overkill and, in fact, putting one up would violate the principle of display inertia (flashing changes on the screen so rapidly that the user cannot keep pace or feels stressed). One could still give less conspicuous progress feedback. A common solution is to combine a "busy" cursor with a rapidly changing number in small field in the bottom of the screen to indicate how much has been done.

See Also:
Alertbox essay about Web response times [http://www.nngroup.com/articles/the-need-for-speed/ and how to improve them.

Question About Web-Based Applications

I keep getting questions like this, so I decided to answer it here.

> **Q:** "You mention many times that response time is important, and there are tons of tools to measure response time, but what is an acceptable web based application's response time? What is a user's tolerance, not for a shopping experience, but for an interactive application?"

A: I wish we could eradicate the term "web-based application" because it distracts from the real issue, which is one of application UI design. We don't have special guidelines for applications implemented in C++ relative to apps implemented in Visual Basic. The fundamental usability recommendations are the same, no matter the implementation, since we are discussing user experience, not coding.

Therefore, the response time guidelines for web-based applications are the same as for all other applications. These guidelines have been the same for 37 years now, so they are also not likely to change with whatever implementation technology comes next.

0.1 second: Limit for users feeling that they are directly manipulating objects in the UI. For example, this is the limit from the time the user selects a column in a table until that column should highlight or otherwise give feedback that it's selected. Ideally, this would also be the response time for sorting the column - if so, users would feel that *they* are sorting the table.

1 second: Limit for users feeling that they are freely navigating the command space without having to unduly wait for the computer. A delay of 0.2-1.0 seconds does mean that users notice the delay and thus feel the computer is "working" on the command, as opposed to having the command be a direct effect of the users' actions. Example: If sorting a table according to the selected column can't be done in 0.1 seconds, it certainly has to be done in 1 second, or users

will feel that the UI is sluggish and will lose the sense of "flow" in performing their task. For delays of more than 1 second, indicate to the user that the computer is working on the problem, for example by changing the shape of the cursor.

10 seconds: Limit for users keeping their attention on the task. Anything slower than 10 seconds needs a percent-done indicator as well as a clearly signposted way for the user to interrupt the operation. Assume that users will need to reorient themselves when they return to the UI after a delay of more than 10 seconds. Delays of longer than 10 seconds are only acceptable during natural breaks in the user's work, for example when switching tasks.

References

Card, S. K., Robertson, G. G., and Mackinlay, J. D. (1991). The information visualizer: An information workspace. *Proc. ACM CHI'91 Conf.* (New Orleans, LA, 28 April-2 May), 181-188.

Miller, R. B. (1968). Response time in man-computer conversational transactions. *Proc. AFIPS Fall Joint Computer Conference* **Vol. 33** , 267-277.

Myers, B. A. (1985). The importance of percent-done progress indicators for computer-human interfaces. *Proc. ACM CHI'85 Conf.* (San Francisco, CA, 14-18 April), 11-17.

Jakob Nielsen's Alertbox: June 21, 2010

Website Response Times

Summary: Slow page rendering today is typically caused by server delays or overly fancy page widgets, not by big images. Users still hate slow sites and don't hesitate telling us.

Users *really* care about speed in interaction design. 13 years ago, I wrote a column called " The Need for Speed," pointing out how much users hated slow-loading Web pages. Back then, big images were the main cause of response-time delays, and our guideline recommended that you keep images small.

Today, most people have broadband, so you might think that download times are no longer a usability concern. And yes, actual **image download is rarely an issue** for today's wireline users (though images can still cause delays on mobile devices).

Still, response times are as relevant as ever. That's because responsiveness is a basic user interface design rule that's dictated by human needs, not by individual technologies. In a client usability study we just completed, for example, users complained that " *it's being a little slow.*"

Speed Matters

Responsiveness matters for two reasons:

- **Human limitations**, especially in the areas of memory and attention (as further discussed in our seminar on The Human Mind and Usability). We simply don't perform as well if we have to wait and suffer the inevitable decay of information stored in short-term memory.

- **Human aspirations**. We like to feel in control of our destiny rather than subjugated to a computer's whims. Also, when companies make us wait instead of providing responsive service, they seem either arrogant or incompetent.

A snappy user experience beats a glamorous one, for the simple reason that people **engage more** with a site when they can move freely and focus on the content instead of on their endless wait.

In a recent study for our work on Brand as Experience, we asked users what they thought about various websites they had used in the past. So, their responses were based not on immediate use (as in normal usability studies), but on whatever *past* experiences were strong enough to form memories. Under these conditions, it was striking to hear users complain about the slowness of certain sites. Slowness (or speed) makes such an impact that it can become one of the brand values customers associate with a site. (Obviously, "sluggish" is not a brand value that any marketing VP would actively aim for, but the actual experience of using a site is more important than slogans or advertising in forming customer impressions of a brand.)

Indeed, we **get findings related to website speed** almost every time we run a study. When sites shave as little as 0.1 seconds off response time, the outcome is a juicy lift in conversion rates. Today or the 1990s? Same effect.

Response-Time Limits

The 3 response-time limits are the same today as when I wrote about them in 1993 (based on 40-year-old research by human factors pioneers):

- **0.1 seconds** gives the feeling of **instantaneous** response — that is, the outcome feels like it was caused by the user, not the computer. This level of responsiveness is essential to support the feeling of **direct manipulation** (direct manipulation is one of the key GUI techniques to increase user engagement and control — for more about it, see our Human Computer Interaction (HCI) for Real World Problems seminar).

- **1 second** keeps the user's flow of thought **seamless**. Users can sense a delay, and thus know the computer is generating the outcome, but they still feel in control of the overall experience and that they're moving freely rather than waiting on the computer. This degree of responsiveness is needed for good navigation.

- **10 seconds** keeps the user's **attention**. From 1–10 seconds, users definitely feel at the mercy of the computer and wish it was faster, but they can handle it. After 10 seconds, they start thinking about other things, making it harder to get their brains back on track once the computer finally does respond.

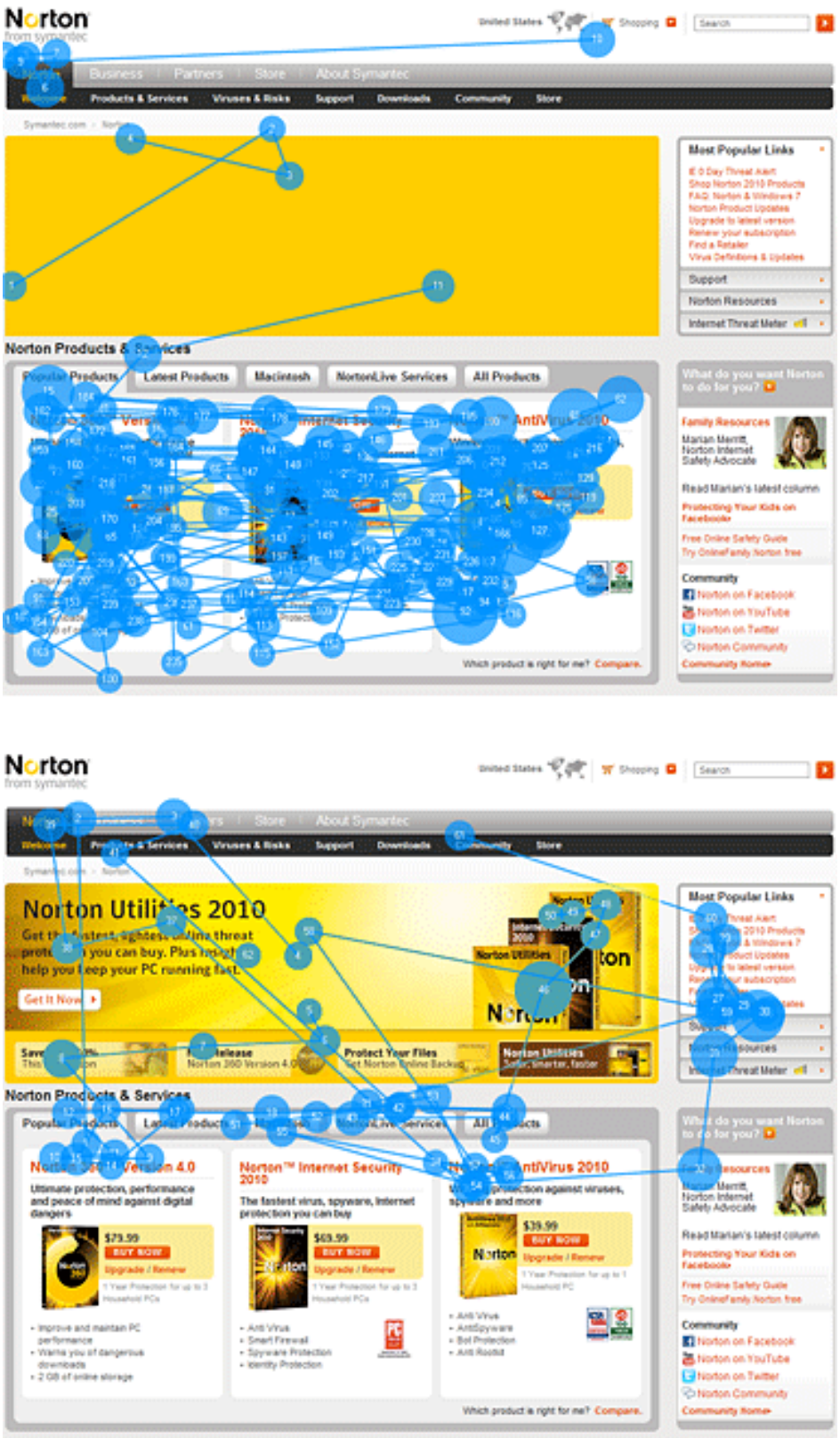
A 10-second delay will often make users **leave a site** immediately. And even if they stay, it's harder for them to understand what's going on, making it less likely that they'll succeed in any difficult tasks.

Even a few seconds' delay is enough to create an **unpleasant** user experience. Users are no longer in control, and they're consciously annoyed by having to wait for the computer. Thus, with repeated short delays, users will give up unless they're extremely committed to completing the task. The result? You can easily lose half your sales (to those less-committed customers) simply because your site is a few seconds too slow for each page.

Fancy Widgets, Sluggish Response

Instead of big images, today's big response-time sinners are typically overly **complex data processing** on the server or **overly fancy widgets** on the page (or *too many* fancy widgets).

Here's an example from a recent eyetracking study we conducted to generate new material for our seminar on Fundamental Guidelines for Web Usability. The following gaze plots show two different users' behavior on the same page, which contained a slideshow widget in the top yellow box that required **8 seconds to download**:



Gaze plots from two different users:
the blue dots indicate where users looked (one fixation per dot).

The test participant in the top gaze plot fixated a few times within the big empty color block before the content downloaded, then spent the remaining time looking at the rest of the page. This user **never looked at the big promotional space** after it had rendered.

The second user (bottom gaze plot) happened to be looking away from the screen during the 8 seconds when the promotional content downloaded. Thus, the first time he looked at the page he saw it as intended, complete with the entire promo.

The slideshow occupies **23% of the page**, not counting a footer that's not shown here. The user who had to endure the download delay spent only **1% of her total viewing time** within this space. In contrast, the user who in effect received instantaneous page rendering (because he didn't look until it was done), spent **20% of his viewing time** within the slideshow area.

Although 8 seconds might not seem like a big delay, it's enough to kill this big promo that the company's Web team probably spent weeks designing. If they had allocated the space to something that rendered in 1 second instead of 8, they would have achieved much better results.

Different Causes, Same Effect

Response times are a matter of user experience: How much time does it take before the computer is ready to serve the user? The reasons behind delays don't matter to users. All they know is that they're getting poor service, which is annoying.

Big images in 1997. Slow servers or overly fancy widgets in 2010. Same effect. Make it snappy, and you'll have a big leg up on the competition and their slow sites.