User Interface Timing Cheatsheet

Revision 0.2.0

Last scrubbed on Sept. 29, 2009

Revision History
Rev 0.1 – Dec 13, 2006

Steve Seow, Ph.D.
sseow@microsoft.com
Microsoft Corp

Contents of this cheatsheet are derived from *Designing and Engineering Time: The Psychology of Time Perception in Software* published by Addison-Wesley. Visit www.EngineeringTime.com for more info.

The 20% Rule

Background:

Research shows that people can start to detect differences between two durations when the percent difference (relative to the shorter duration) exceeds 20%.

Rule of Thumb:

- Ensure that percentage of delta between two durations exceeds 20% of shorter to increase chance of users detecting difference.
- Conversely, ensure that percentage is less than 20% if you do not want users to detect difference.

Mini Case Studies:

- Newly proposed compression algorithm reduces download from 11 seconds to 10 seconds—should we go ahead and invest in this change?

Verdict: May not be worth the investment because users may not detect difference.

- A proposed security enhancement to a feature will Add an additional 3 seconds to the 20 second load time. Is it safe to implement this feature? **Verdict:** Still within the safe window because this is approximately 15% increase, users may not detect the difference.

Responsiveness: Instantaneous behavior

Background:

Physical objects observe physical laws. Software UI objects observe whatever law we define in code. Research and industry standards point to 0.1 to 0.2 seconds as the range of maximum acceptable response time to simulate instantaneous behavior.

Rule of Thumb:

- Any UI that mimics objects in the real world (button, scrollbars, etc.) must be coded to exhibit instantaneous behavior.
- User-centric response time is measured with user implicit action to user-detectable system response. Add 50 ms to timing reported by code.

Mini Case Study:

- A button on a website is taking 0.4 seconds from mouse-click to the playback of a click sound. **Verdict:** Because the click sound is mimicking a sound coming from the pressing of a mechanical button, its timing is not adequate, and must be reduced at least by half.

Responsiveness: Immediate Response

Background:
Research and industry standards point to a 0.5 to 1 second as the maximum acceptable response time for immediate system responses, as in a signal or acknowledgement to the user that a command or instruction has been received.

Rule of Thumb:

- Processes perceived by users as easily performed (page down, zooming in and out, maximizing and minimizing of windows, etc.) must be executed and completed in or before the Immediacy window.
- If info requested by users is perceived to be 'ready', such as some text on a webpage that falls outside of the viewable space on a browser, the expectation is that what is requested is "already there" and getting to it is merely navigating or panning it into view.

Mini Case Studies:

- A web search returns a total of 10,000 results and the first 10 are shown. Users click a button to view the next 10 results. The second set of results was displayed in 4 seconds. **Verdict:** Because users has been given the indication that the search has already completed and that viewing the remaining results are merely bringing them to view, the response time must be reduced substantially.
- User double-clicks on an already displayed small file to see its code. The development tool takes 4

seconds to fully load the file as a tab document. **Verdict:** Because the file perceived by the user as already loaded and relatively light-weight, 4 seconds far exceeds the Immediacy window.

Responsiveness: User Flow

Background:

A smooth user flow, where users feel that they are making continuous progress, is essential to the overall experience. Research and industry standards points to 2 to 5 seconds as the range of maximum acceptable response time for ensuring a continuous user flow. Ideally, the system must provide some indication of progress or results within 2 seconds for simple instructions and at most by 5 seconds if the process is perceived as a complex task.

Rule of Thumb:

- Very short waits under 2 seconds can do without details and progress bars. Simple animation or indication (wait cursors, etc.) in the UI to indicate some work is being done is sufficient.
- Short waits above 2 seconds must begin to show progress feedback (progress bars) which allows users to eyeball the amount of progress.
- Waits above 5 seconds *must* begin to show more details such as time remaining or percent completed.
- To maintain continuity, provide status updates every 2 to 5 seconds.

Mini Case Study:

- An application needs to reach a resource on the Internet each time it is launched. Since the time it takes to reach the resource depends on many unpredictable factors, the application was designed to wait until 20 seconds before timing out and informing the user that resource is not available. **Verdict:** 20 seconds far too long to maintain continuity. The application ought to timeout within 5 seconds or provide an indication every 2 to 5 seconds that it is still attempting connection.

Captivity: Attention Span

Background:

Numerous lines of research show that most users have an attention span of 7 to 10 seconds. This is best understood as the system's opportunity to make an "elevator pitch" to the user to keep the user interested.

Rule of Thumb:

- Do not hold users captive to the wait over 10 seconds. If a process needs to make users wait beyond the captivity limit, allow users to cancel or task out of the wait.
- Ensure that users get useful and consumable information within 7 to 10 seconds.

Mini Case Study:

- An application was coded to invite first-time users to fill out an Internet survey when the application is launched for the first time. Upon clicking the link, the average time taken for the survey site to fully load is about 12 seconds. **Verdict:** The internet survey, made worse by the fact that it is voluntary, is exceeding the average attention span. It would be wise to reduce down to 7 seconds or under.

Expressing Time in UI: Time Anchors

Background:

For time estimations under a few hours, people tend to gravitate towards 1, 2, 3, 5, 10, 15, 20, and 30. (The numbers 4 and 45 are close contenders as well). That is, when asked to estimate a short duration, people are prone to using one or more numbers, such as 10 seconds or 2 to 3 minutes, to describe the duration. This is observable for durations in the magnitude of hours, but to a lesser extent. These time anchors can be represented in what is termed here as the Time Anchor Matrix:

1	2	3
5	10	15
10	20	30

Rule of Thumb:

- See *Expressing Ranges*
- See *Expressing Upper Limits*
- See *Countdowns*

Expressing Ranges: 'Between X to Y'

Background:

Time anchors come in handy when there is a need to specify a range of times to represent the possible durations of an event. This happens frequently when one or more other factors can influence the variability of the duration. For example, if we are confident that a particular process will take around 4 minutes, we can state (display in the UI) a range that spans over 4 minutes. Referring to the matrix, we see that 3 and 5 are the integers that are the two candidates and as such, we state that the process will take between 3 and 5 minutes.

Rule of Thumb:

- Do not to skip over any successive time anchor in expressing the two ends of a range. For example, we don't want to state 5 to 15 minutes because they skip over 10.

Examples:

- A progress indicator is showing that a particular process will be completed between 4 to 20 minutes. **Verdict:** This is not a good practice. When time anchors are not used, people get the impression that the range is too wide and thus unreasonable.

Upper Limits: 'Less than X'

Background:

When it is not possible or wise to state a range, we may resort to stating an upper limit or the longest possible duration in a statement like 'less than 5 minutes'.

Rule of Thumb:

- Round up to the next element in the Time Anchor Matrix in stating upper limits. For example, if we are confident that a particular process will be completed in 7 minutes and 50 seconds, we'll state that it process will take under 10 minutes.
- As far as UI design is concerned, there are few advantages in stating lower limits, such as 'This download will take at least 5 minutes', or 'This will take more than 5 minutes."

Mini Case Study:

- The UI of a compiling process is stating that it will finish in 3 minutes and 25 seconds. **Verdict:** Unless there is a strong reason to be precise, this is not a good practice. When we use numbers that users are not accustomed to using, (that is, numbers not represented by the Time Anchor Matrix), we risk making them hold us to what we promised. This expectation is likely due to the fact that your statement is construed as one that has been made after rigorous timing and performances testing that have somehow locked down 3 minutes and 25 seconds with precision.

Countdowns: ‘X-3, X-2, X-1, X’

Background:

Unless there is a compelling need to use time-elapsed timer (0:05… 0:06… 0:07…), it is always better to use a time-remaining or count-down timer (0:04… 0:03… 0:02…) if there is any need to show a timer at all.

Rule of Thumb

- Reserve such use of countdown timers to relatively shorter durations that are around 10 minutes and shorter.
- Use time anchors for countdown timers. Instead of specifying every single second between 10 minutes and 0 second (10:00… 9:59… 9:59… 9:58…), for example, express the countdown in time anchor units: (10 minutes… 5 minutes… 3 minutes… 2 minutes… 1 minute… 30 seconds…)
- Ensure that the time units are appropriately singular when necessary. For example, 1 minute not 1 minute*s*, 0 second not 0 second*s*.

Mini Case Study:

- A DVD burning application is showing that the process will complete in 5:32, 5:31, 5:30… **Verdict:** This is not a good practice. Change the units to time anchors: 5:30 >5:15 >5:00…