

We Have Time Requirements

Events in the world take time to play out. *Perceiving* objects and events also takes time, as does remembering perceived events, thinking about past and future events, learning from events, acting on plans, and reacting to perceived and remembered events. How much time? And how does knowing the duration of perceptual and cognitive processes help us design interactive systems?

This chapter answers those questions. It presents the *durations* of perceptual and cognitive processes and based on those provides real-time *deadlines* that interactive systems must meet to synchronize well with human users. Interactive systems that don't synchronize well with users' time requirements are (1) less effective and (2) perceived by users as unresponsive.

Issue (2), perceived responsiveness, may seem less important than effectiveness, but in fact it is *more* important. Over the past 5 decades, researchers have found consistently that an interactive system's responsiveness—its ability to keep up with users, keep them informed about its status, and not make them wait unexpectedly—is a *very* important factor in determining user satisfaction.¹ This has been repeatedly confirmed by research conducted over many decades (Miller, 1968; Thadhani, 1981; Barber and Lucas, 1983; Carroll and Rosson, 1984; Shneiderman, 1984; Lambert, 1984; Rushinek and Rushinek, 1986; Hoxmeier and DiCesare, 2000; Nah, 2004). Prominent software design experts also recommend ensuring that apps and websites respond quickly enough to meet human time requirements (Nielsen, 1993, 1997, 2010, 2014; Isakson, 2013; Heusser, 2019).

This chapter first defines responsiveness. It then enumerates important time constants of human perception and cognition. It ends with real-time guidelines for interactive system design, including examples.

¹Some researchers have suggested that for users' perception of the loading speed of websites, the causality may go the other way: the more success people have at a site, the faster they think it is, even when their ratings don't correlate with the sites' actual speed (Perfetti and Landesman, 2001).

RESPONSIVENESS DEFINED

Responsiveness is related to performance but is different. Performance is measured in terms of computations per unit of time. Responsiveness is measured in terms of compliance with human time requirements and user satisfaction.

Interactive systems can be responsive despite low performance. If you call a friend to ask a question, she can be responsive even if she can't answer your question immediately: she can acknowledge the question and promise to call back. She can be even more responsive by saying *when* she will call back.

Responsive systems keep a user informed even if they cannot fulfill the user's requests immediately. They provide feedback about what the user has done and what is happening, and they prioritize the feedback based on *human* perceptual, motor, and cognitive deadlines (Duis and Johnson, 1990). Specifically, they:

- let you know immediately that your input was received;
- provide some indication of how long operations will take (see Fig. 14.1);
- free you to do other things while waiting;
- manage queued events intelligently;
- perform housekeeping and low-priority tasks in the background;
- anticipate your most common requests.

Software can have poor responsiveness even if it is fast. Even if a camera repairman is very fast at fixing cameras, he is unresponsive if you walk into his shop and he ignores you until he finishes working on another camera. He is unresponsive if you hand him your camera and he silently walks away without saying whether he is going to fix it now or go to lunch. Even if he starts working on your camera immediately, he is unresponsive if he doesn't tell you whether fixing it will take 5 minutes, 5 hours, 5 days, or 5 weeks.

Systems that display poor responsiveness do not meet human time deadlines. They don't keep up with users. They don't provide timely feedback for user actions, so users are unsure of what they have done or what the system is doing. They make users

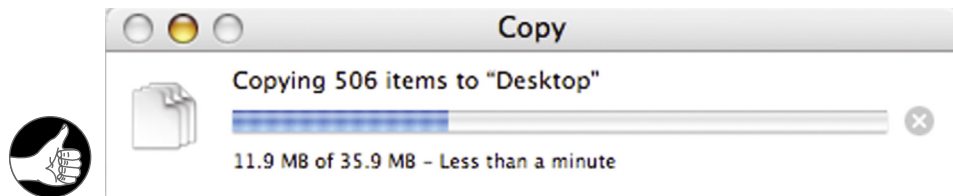


FIGURE 14.1

MacOS X file transfer: good progress indicator, useful time estimate, and cancel button.

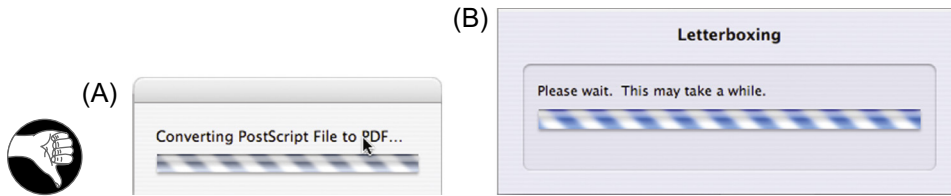


FIGURE 14.2

No progress bar (just a busy bar) and no cancel for (A) macOS X and (B) iMovie.

wait at unpredictable times and for unpredictable periods. They limit users' work pace—sometimes severely. Here are some specific examples of poor responsiveness:

- delayed feedback for button presses, scrollbar movement, or object manipulations
- time-consuming operations that block other activity and cannot be aborted (see [Fig. 14.2A](#))
- providing no clue how long lengthy operations will take (see [Fig. 14.2B](#))
- jerky, hard-to-follow animations
- ignoring user input while performing “housekeeping” tasks users did not request

These problems impede users' productivity and frustrate and annoy them. Unfortunately, despite all the research showing that responsiveness is critical to user satisfaction and productivity, many of today's interactive systems have poor responsiveness (Johnson, 2007).

THE MANY TIME CONSTANTS OF THE HUMAN BRAIN

To understand the time requirements of human users of interactive systems, we will start with neurophysiology.

The human brain and nervous system are not really a single organ; rather, they are made up of a collection of neuron-based organs that appeared at vastly different points in evolution (see [Chapter 10](#)). This collection provides a large variety of sensory, regulatory, motor, and cognitive functions. Not surprisingly, these functions operate at different speeds. Some work very fast, executing functions in small fractions of a second, while others are many, many times slower, executing over many seconds, minutes, hours, or even longer time spans.

Our different sensory systems operate at different speeds. For example, the brain processes auditory and tactile input faster than visual input. But when we clap our hands, we perceive the sound, touch, and sight of the clap as simultaneous. The brain waits for input from all our senses, then edits the timeline to synchronize input from our ears, hands, and eyes (Eagleman, 2015). If that suggests to you that your conscious self lives a fraction of a second in the past, you are right; it does.

A second example of the brain’s multiple operating speeds is the difference between automatic processing (system one) and controlled processing (system two).² Automatic processing, such as playing a memorized musical piece, operates on a “clock” at least 10 times faster than the one governing highly monitored, controlled processing, such as *composing* a musical piece.

A third example is the flinch reflex. A region of the brain called the superior colliculus—part of the evolutionarily ancient old brain—can “see” a rapidly approaching object and make you flinch or raise your arms to protect yourself long before your cerebral cortex—part of the new brain—has perceived and identified the object. Unlike the cortex, the superior colliculus cannot identify objects. But it does not need to; it just has to activate your body’s reflexes to defend itself against rapidly approaching objects.

The sidebar “How long does the brain take to ... ?” gives the durations of some important perceptual and cognitive brain functions. Most are self-explanatory, but a few require additional explanation.

HOW LONG DOES THE BRAIN TAKE TO ... ?

Table 14.1 shows durations of perceptual and cognitive functions that affect our perceptions of system responsiveness, from shortest to longest (Card et al., 1991; Johnson, 2007; Sousa, 2005; Stafford and Webb, 2005). Those that are not self-explanatory are explained more fully later.

Table 14.1 Durations of Perceptual and Cognitive Functions

Perceptual and Cognitive Functions	Duration
Shortest gap of silence that we can detect in a sound	1 millisecond (0.001 second)
Minimum time between spikes in auditory neurons, the fastest neurons in the brain	2 milliseconds (0.002 second)
Shortest time a visual stimulus can be shown and still affect us (perhaps unconsciously)	5 milliseconds (0.005 second)
Minimum noticeable lag in ink as someone draws with a stylus	10 milliseconds (0.01 second)
Maximum interval for auditory fusion of successive sound pulses into a pitched tone	20 milliseconds (0.02 second)
Maximum interval for visual fusion of successive images	50 milliseconds (0.05 second)
Speed of flinch reflex (involuntary motor response to possible danger)	80 milliseconds (0.08 second)
Time lag between a visual event and our full perception of it (or perceptual cycle time)	100 milliseconds (0.1 second)
Threshold for perceptual “locking” of events and sounds	100 milliseconds (0.1 second)
Duration of saccade (involuntary eye movement), during which vision is suppressed	100 milliseconds (0.1 second)

²See Chapter 10.

Table 14.1 Durations of Perceptual and Cognitive Functions—cont'd

Perceptual and Cognitive Functions	Duration
Maximum interval between events for perception that one event caused another event	140 milliseconds (0.14 second)
Time for a skilled reader's brain to comprehend a printed word	150 milliseconds (0.15 second)
Time to subitize (determine the number of) four to five items in our visual field	200 milliseconds (0.2 second; 50 milliseconds/item)
Editorial "window" within which the brain edits the presence and order of events	200 milliseconds (0.2 second)
Time to identify (i.e., name) a visually presented object	250 milliseconds (0.25 second)
Time to mentally count each item in a scene when there are more than four items	300 milliseconds (0.3 second)
Attentional "blink" (inattentiveness to other input) following recognition of an object	500 milliseconds (0.5 second)
Visual-motor reaction time (intentional response to unexpected event)	700 milliseconds (0.7 second)
Maximum duration of silent gap between turns in person-to-person conversation	About 1 second
Duration of unbroken attention to a single task (unit task)	6–30 seconds
Time to make critical decisions in emergency situations (e.g., medical triage)	1–5 minutes
Duration of important purchase decision (e.g., buying a car)	1–10 days
Time to choose a lifetime career	20 years

Shortest gap of silence that we can detect in a sound: 1 millisecond (0.001 second)

Our hearing is much more sensitive to short events and small differences than our vision is. Our ears operate using mechanical sound transducers, not electrochemical neural circuitry. The eardrum transmits vibration to the ossicles (middle-ear bones), which in turn transmit vibration to the cochlea's hair cells, which when vibrated trigger electrical pulses that go to the brain. Because the connection is mechanical, our ears respond to sound much faster than the rods and cones in our retina respond to light. This speed allows our auditory system to detect very small differences in the time when a sound arrives at our two different ears, from which the brain calculates the direction of the sound's source.

Shortest time a visual stimulus can be shown and still affect us (perhaps unconsciously): 5 milliseconds (0.005 second)

This is the basis of so-called subliminal perception. If you are shown an image for 5–10 milliseconds, you won't be aware of seeing it, but low-level parts of your visual system

will register it. One effect of such exposure to an image is that your familiarity with it will increase; if you see it again later, it will seem familiar. Brief exposure to an image or a looming object can also trigger responses from your old brain—avoidance, fear, anger, sadness, joy—even if the image disappears before the conscious mind identifies it. However, contrary to popular myth, subliminal perception is not a strong determinant of behavior. It cannot make you do things you wouldn't otherwise do or want things you wouldn't otherwise want (Stafford and Webb, 2005).

Speed of flinch reflex (involuntary motor response to possible danger): 80 milliseconds (0.08 second)

When an object—even a shadow—approaches you rapidly, you hear a loud sound nearby, or something suddenly pushes, jabs, or grabs you, your reflex is to flinch: pull away, close your eyes, throw up your hands in defense, etc. This is the flinch reflex. It is very fast compared with intentional reaction to a perceived event—about 10 times faster. Evidence of the speed of the flinch reflex has been seen not only experimentally but also in examining the injuries that occur when people are attacked or involved in vehicle accidents; often their arms and hands are injured in ways that indicate that they managed to get their hands up in a split second (Blauer, 2007).

Time lag between a visual event and our full perception of it (or perceptual cycle time): 100 milliseconds (0.1 second)

From the time that light from an external event hits your retina to the time that neural impulses from that event reach your cerebral cortex, about 0.1 second elapses. Suppose our conscious awareness of the world lagged behind the real world by 1/10th of a second. That lag would not be conducive to our survival: 1/10th of a second is a long time when a rabbit you are hunting darts across a meadow. The brain compensates by extrapolating the position of moving objects by 0.1 second. Therefore, as a rabbit runs across your visual field, you see it where your brain estimates it is *now*, not where it was 0.1 second ago (Stafford and Webb, 2005). However, your brain also edits the order of events before you are aware of them (see explanation of editorial window, below), so if the running rabbit suddenly turns left, your extrapolating brain doesn't falsely make you see it continuing straight ahead and then have to backtrack.

Threshold for perceptual “locking” of events and sounds: 100 milliseconds (0.1 second)

The brain “locks” a visual event to a corresponding sound if the delay between them is less than 0.1 second. If you watch a man beat a drum from several hundred meters away, you first see him hit the drum and then hear the sound. The closer you are to the drummer, the shorter the delay. However, if the delay goes under 100 milliseconds (which it does at a distance of about 33.5 m, or 100 ft), the brain “locks” the visual perception to the auditory one, and you no longer notice the delay. This fact means that editors of television shows, movies, video games, and other visual media can allow

up to 100 milliseconds of “slop” in synchronizing visuals with audio before people notice (Bilger, 2011; Eagleman, 2015).


Duration of saccade (involuntary eye movement), during which vision is suppressed: 100 milliseconds (0.1 second)

As described in [Chapter 6](#), our eyes move frequently, and to a great extent involuntarily, about three times per second. These are called saccadic eye movements, or *saccades*. Each saccade takes about 1/10th of a second, during which vision is suppressed (i.e., turned off). This brief shutdown of vision is called *saccadic masking*. However, we don’t notice these blank intervals. The brain edits out the blank intervals and stitches together the visual clips before they reach our awareness, making it seem to us as if they didn’t exist. You can demonstrate this to yourself by facing a mirror and looking at your left eye, then your right eye, and so on. You don’t see blank intervals while your eyes move, and you also never see your eyes move. It’s as if the movement took no time. However, someone else watching you will see your eyes moving back and forth, taking a noticeable amount of time to move from left to right (Bilger, 2011; Eagleman, 2012).

Maximum interval between events for perception that one event caused another event: 140 milliseconds (0.14 second)

This interval is the deadline for perception of cause and effect. If an interactive system takes longer than 0.14 second to respond to your action, you won’t perceive your action as having caused the response. For example, if the echoing of characters you type lags more than 140 milliseconds behind your typing, then you will lose the perception that you are typing those characters. Your attention will be diverted away from the meaning of the text and toward the act of typing, which slows you down, pulls typing out of automatic processing and into conscious processing, and increases your chances of making an error.

Time to subitize (determine the number of) four to five items in our visual field: 200 milliseconds (0.2 second; 50 milliseconds/item)

If someone tosses two coins onto the table and asks how many coins there are, it takes only a glance for you to see that there are two. You don’t have to explicitly count them. You can do the same with three coins, or four. Some people can do it with five. This function is called *subitizing*. Beyond four or five, it gets harder. Now you are starting to have to count, or if the coins happen to fall into separate groups on the table, you can subitize each subgroup and add the results. This phenomenon is why when we count objects using tick-marks, we write the tick-marks in groups of four, then draw the fifth tick-mark across the group, like this: . Subitizing feels instantaneous, but it isn’t. It takes about 50 milliseconds per item (Card et al., 1983; Stafford and Webb, 2005). However, that is much less time per item than explicit counting, which takes about 300 milliseconds per item.

Editorial “window” within which the brain edits the presence and order of events: 200 milliseconds (0.2 second)

The order in which we perceive events is not necessarily the order in which they occur. The brain has a moving “editorial window” of about 200 milliseconds during which perceived and recalled items vie for promotion to consciousness. Within that time window, events and objects that might otherwise have made it to consciousness may be superseded by others—even ones that occurred later in time (within the window). Within the window, events can also be resequenced on the way to consciousness. An example: We see a dot that disappears and immediately reappears in a new position as moving. Why? Our brain certainly does not do it by “guessing” the second dot’s position and showing us “phantom” motion in that direction, because we see motion in the correct direction *regardless* of where the new object appears. The answer is that we don’t actually perceive motion until the dot appears in the new position, but the brain resequences things so that we seem to see the motion *before* the dot reappears. The second dot must appear less than 0.2 second after the first dot disappears for the brain to be able to resequence the events (Stafford and Webb, 2005; Eagleman, 2012, 2015).

Attentional “blink” (inattentiveness to other objects) following recognition of an object: 500 milliseconds (0.5 second)

As described in [Chapter 1](#), this is one way in which our perception is biased. Briefly, if you spot or hear something or someone that captures your attention, you are essentially deaf and blind to other events for about a half-second. Your brain is in “busy” mode.

With a colleague’s help, you can demonstrate the “blink.” Choose two words. Tell the colleague the two words. Then explain that you will read a list of words and at the end you want to know if either of the two words was in the list. Quickly read a long list of words at a rate of three words per second. Somewhere in the list, include one target word. If the second target word is presented right after the first—within one or two items—your colleague probably won’t hear it.

Visual–motor reaction time (intentional response to unexpected event): 700 milliseconds (0.7 second)

This interval is the combined time for your visual system to notice something in the environment and initiate a conscious motor action and for the motor system to execute that action. If you are driving your car toward an intersection and the light turns red, this is the time required for you to notice the red light, decide to stop, and put your foot on the brake pedal. How long it takes your car to actually *stop* is not included in the 700 milliseconds. The vehicle stopping time depends on how fast the car is going, the condition of the pavement under the wheels, etc.

This reaction time is different from the flinch reflex—the old brain responding to rapidly approaching objects, making you automatically close your eyes, dodge, or

throw up your hands to protect yourself. That reflex operates about 10 times faster (see earlier).

The visual-motor intentional reaction time is approximate. It varies a bit among people. It also increases with distractions, drowsiness, blood-alcohol level, and possibly age.

Maximum duration of silent gap between turns in person-to-person conversation: about 1 second

This is the approximate normal length of gaps in a conversation. When gaps exceed this limit, participants—either speakers or listeners—often say something to keep the conversation going: they interject “uh” or “uh-huh” or take over as speaker. Listeners respond to such pauses by turning their attention to the speaker to see what caused it. The precise length of such gaps varies by culture, but it is always in the range of 0.5–2 seconds.

Duration of unbroken attention to a single task (“unit task”): 6–30 seconds

When people perform a task, they break it down into little pieces: subtasks. For example, buying airline tickets online consists of: (1) going to a travel or airline website, (2) entering the trip details, (3) scanning the results, (4) selecting a set of flights, (5) providing credit card information, (6) reviewing the purchase, and (7) finalizing the purchase. Some subtasks are broken down further, for example, entering the trip details consists of entering the trip origin, destination, dates, time, etc., piece by piece. This breaking down of tasks into subtasks ends with small subtasks that can be completed without a break in concentration, with the subgoal and all necessary information either held in working memory or directly perceivable in the environment. These bottom-level subtasks are called *unit tasks* (Card et al., 1983). Between unit tasks, people typically look up from their work, check to see if anything else requires attention, perhaps look out the window or take a sip of coffee, etc. Unit tasks have been observed in activities as diverse as editing documents, entering checkbook transactions, designing electronic circuits, and maneuvering fighter jet planes in dogfights, and they always last somewhere in the range of 6 to 30 seconds.

ENGINEERING APPROXIMATIONS OF TIME CONSTANTS: ORDERS OF MAGNITUDE

Interactive systems should be designed to meet the temporal requirements of their human users. However, trying to design interactive systems for the wide variety of perceptual and cognitive time constants would be nearly impossible.

But people who design interactive systems are engineers, not scientists. We don’t have to account for the full variety of brain-related time constants and clock-cycle times. We just

have to design interactive systems that work for human beings. This more approximate requirement gives us the freedom to consolidate the many perceptual and cognitive time constants into a smaller set that is easier to teach, remember, and use in design.

Examining the list of critical durations presented in [Table 14.1](#) yields some useful groupings. Times related to sound perception are all on the order of a millisecond, so we can consolidate them all into that value. Whether they are really 1 millisecond, or 2, or 3—we don't care. We only care about factors of 10.

Similarly, there are groups of durations around 1, 10, 100 milliseconds, 1, 10, and 100 seconds. Above 100 seconds, we are beyond durations that most interaction designers care about. Thus, for designing interactive systems, these consolidated deadlines provide the required accuracy.

Notice that these deadlines form a convenient series: each successive deadline is 10 times—one order of magnitude—greater than the previous one. That makes the series fairly easy for designers to remember, although remembering what each deadline represents may be challenging.

DESIGNING TO MEET REAL-TIME HUMAN-INTERACTION DEADLINES

To be perceived by users as responsive, interactive software must follow these guidelines:

- Acknowledge user actions instantly even if returning the answer will take time; preserve users' perceptions of cause and effect.
- Let users know when the software is busy and when it is not.
- Free users to do other things while waiting for a function to finish.
- Animate movement smoothly and clearly.
- Allow users to abort (cancel) lengthy operations they don't want.
- Allow users to judge how much time lengthy operations will take.
- The software should do its best to let users set their own work pace.

In these guidelines, *instantly* means within about 0.1 second. Much longer than that, and the user interface will have moved out of the realm of cause and effect, reflexes, perceptual-motor feedback, and automatic behavior into the realm of conversational gaps and intentional behavior (see sidebar: “How long does the brain take to ... ?”). After 2 seconds, a system has exceeded the expected time for turn-taking in dialogue and has moved into the time range of unit tasks, decision-making, and planning.

Now that we have listed time constants of human perception and cognition and consolidated them into a simplified set, we can quantify terms such as *instantly*, *take time*, *smoothly*, and *lengthy* in the preceding guidelines (see also [Table 14.2](#)).

Table 14.2 Time Deadlines for Human–Computer Interaction

Deadline (Seconds)	Perceptual and Cognitive Functions	Deadlines for Interactive System Design
0.001	<ul style="list-style-type: none"> • minimum detectable silent audio gap 	<ul style="list-style-type: none"> • maximum tolerable delay or dropout time for audio feedback (e.g., tones, “earcons,” music)
0.01	<ul style="list-style-type: none"> • preconscious perception • shortest noticeable pen-ink lag 	<ul style="list-style-type: none"> • inducing unconscious familiarity of images or symbols • generating tones of various pitches • electronic ink maximum lag time
0.1	<ul style="list-style-type: none"> • subitizing (perceiving the number of) one to four items • involuntary eye movement (saccade) • audiovisual “lock” threshold • flinch reflex • perception of cause and effect • perceptual–motor feedback • visual fusion • object identification • editorial window of consciousness • the perceptual “moment” 	<ul style="list-style-type: none"> • assume users can “count” one to four screen items in ~100 milliseconds, but more than four takes 300 milliseconds/item • feedback for successful hand–eye coordination (e.g., pointer movement, object movement, or resizing, scrolling, drawing with mouse) • tolerable “slop” in synching sound with visual events • feedback for click on button or link • displaying “busy” indicators • allowable overlap between speech utterances • max interval between animation frames
1	<ul style="list-style-type: none"> • max delay for continuity of thought • max delay for effective navigation • max conversational gaps • visual–motor reaction time for unexpected events • attentional “blink” 	<ul style="list-style-type: none"> • displaying progress indicators for long operations • finishing user-requested operations (e.g., open window) • finishing unrequested operations (e.g., autosave) • time after information presentation that can be used for other computations (e.g., to make inactive objects active) • required wait time after presenting important information before presenting more
10	<ul style="list-style-type: none"> • unbroken concentration on a task • unit task: one part of a larger task 	<ul style="list-style-type: none"> • completing one step of a multistep task (e.g., one edit in a text editor) • completing user input to an operation • completing one step in a wizard (multistep dialogue box)
100	<ul style="list-style-type: none"> • critical decision in emergency situation 	<ul style="list-style-type: none"> • assure that all information required for a decision is provided or can be found within this time

0.001 second (1 millisecond)

As described earlier, the human auditory system is sensitive to very brief intervals between sounds. If an interactive system provides audible feedback or content, its audio-generation software should be engineered to avoid network bottlenecks, getting swapped out, deadlocks, and other interruptions. Otherwise, it may produce

noticeable gaps, clicks, or lack of synchrony between audio tracks. Audio feedback and content should be provided by well-timed processes running with high priority and sufficient resources.

0.01 second (10 milliseconds)

Subliminal perception is rarely if ever used in interactive systems, so we needn't concern ourselves with that issue. Suffice it to say that if designers wanted to boost the familiarity of certain visual symbols or images without user awareness, they could do so by presenting the images or symbols repeatedly for 10 milliseconds at a time. It is also worth mentioning that while extremely brief exposure to an image can increase its familiarity, the effect is weak—certainly not strong enough to make people like or dislike specific products.

One way for software to generate tones is by sounding clicks at various rates. If the clicks are less than 10 milliseconds apart, they will be heard as a single sustained buzz in which the pitch is determined in part by click frequency. Users will hear clicks as distinct if they are separated by intervals of more than 10 milliseconds.

Systems in which users write with a finger or stylus should ensure that the electronic “ink” does not lag behind by more than 10 milliseconds; otherwise, users will notice the lag and be annoyed.

0.1 second (100 milliseconds)

If software waits longer than 0.1 second to show a response to a user's action, the perception of cause and effect is broken; the software's reaction will not seem to be a result of the user's action. Meeting the 0.1 second deadline is essential to support users' perceptions that they are *directly manipulating* objects on the display (Nielsen, 2010). Therefore, onscreen buttons have 0.1 second to show they've been clicked; otherwise, users will assume they missed and click again. This does not mean that buttons have to complete their *function* in 0.1 second—only that they must show that they have been *pressed* by that deadline.

The main design point about the flinch reflex is that interactive systems should not startle users and cause flinching. Other than that, the flinch reflex and its duration don't seem very relevant to interactive system design. It is difficult to imagine beneficial uses of the flinch reflex in human-computer interaction, but one can imagine games with loud noises, joysticks with sudden tactile jolts, or three-dimensional virtual environments that cause their users to flinch under some circumstances, perhaps purposefully. For example, if a vehicle detects a pending collision, it could do something to make riders flinch to help protect them upon impact.

If an object the user is dragging or resizing lags more than 0.1 second behind users' pointer movements, they will have trouble placing or resizing the object as desired. Therefore, interactive systems should prioritize feedback for hand-eye coordination tasks so that the feedback never lags past this deadline. If meeting that deadline is unachievable, the system should be designed so the task does not require close hand-eye coordination.

If an operation will take more than a perceptual “moment” (0.1 second) to complete, it should display a busy indicator. If a busy indicator can be displayed within 0.1 second, it can double as the action acknowledgment. If not, the software’s response should come in two parts: a quick acknowledgment within 0.1 second, followed by a busy (or progress) indicator within 1 second. More guidance for displaying busy indicators is given later.

The brain can reorder events within this approximate time window before the events reach consciousness. Human speech is highly prone to such reordering if it occurs out of order. If you listen to several people talking and some people start talking just before the person before them has finished talking (within the time window), your brain automatically “untangles” the utterances so that you seem to hear them sequentially without perceived overlap. Television and movies sometimes take advantage of this phenomenon to speed up conversations that normally would take too long.

We also regard 10 frames per second as an approximate minimum frame rate for perception of smooth animation, even though smooth animation really requires a rate more like 20 frames per second.

1 second

Computer systems that respond within 1 second allow users to maintain a seamless, continuous thought process as they work or navigate toward a goal (Nielsen, 2010).

Because 1 second is the maximum gap expected in conversation, and because operating an interactive system is a form of conversation, interactive systems should avoid lengthy gaps in their side of the conversation. Otherwise, the human user will wonder what is happening. Systems have about 1 second to either do what the user asked or indicate how long it will take. Otherwise, users get impatient.

If an operation will take more than a few seconds, a progress indicator is needed. Progress indicators are an interactive system’s way of keeping its side of the expected conversational protocol: “I’m working on the problem. Here is how much progress I have made and an indication of how much more time it will take.” More guidelines for progress indicators are provided later.

One second is also the approximate minimum time a user needs to respond intentionally to an unanticipated event. Therefore, when information suddenly appears on the screen, designers can assume that users will take at least 1 second to react to it (unless it causes a flinch response; see earlier discussion). That lag time can be useful in cases when the system needs to display an interactive object but cannot both render the object and make it interactive within 0.1 second. Instead, the system can display a “fake,” inactive version of the object, and then take its time (1 second) to fill in details and make the object fully interactive. Computers can do a lot in 1 second.

10 seconds

Ten seconds is the approximate time limit of human attention—i.e., short-term memory—unless something happens to refresh the thought. Therefore, it is the approximate duration of a “unit task”—the unit of time people usually use to break down

their planning and execution of larger tasks. Interruptions of more than 10 seconds cause users' minds to wander, and when the computer eventually does respond, users must somehow bring their thoughts back to what they were doing (Nielsen, 2010).

Examples of unit tasks are completing a single edit in a text-editing application, entering a transaction into a bank account program, and executing a maneuver in an airplane dogfight. Software should support segmentation of tasks into these 10-second pieces.

Ten seconds is also roughly the amount of time users are willing to spend setting up “heavyweight” operations like file transfers or searches—if it takes any longer, users start to lose patience. Computing the result can then take longer if the system provides progress feedback.

Similarly, each step in a multipage “wizard” dialogue box should have at most about 10 seconds of work for a user to do. If one step of the wizard takes significantly longer than 10 seconds to complete, it probably should be broken up into multiple smaller steps.

100 seconds (~1.5 minutes)

Interactive systems that support rapid critical decision-making should be designed so that all necessary information either is already in front of the decision-maker or can be easily obtained with minimal browsing or searching. The best user interface for this type of situation is one in which users can obtain all crucial information simply by moving their eyes to where it is displayed³ (Isaacs and Walendowski, 2001).

ADDITIONAL GUIDELINES FOR ACHIEVING RESPONSIVE INTERACTIVE SYSTEMS

In addition to design guidelines specific to each consolidated human-computer interaction deadline, there are general guidelines for achieving responsiveness in interactive systems.

Use busy indicators

Busy indicators vary in sophistication. At the low end, we have simple static wait-cursors (e.g., an hourglass). They provide very little information, only that the software is temporarily occupied and unavailable to the user for other operations.

Next, we have wait-animations. Some of these are animated wait-cursors, such as the macOS rotating color wheel. Some wait-animations are not cursors but rather larger graphics elsewhere on the screen, such as the “downloading data” animations displayed by some web browsers. Wait-animations are more user-friendly than static wait-cursors because they show that the system is working, not crashed or hung up

³ sometimes called “no-click” user interfaces.

waiting for a network connection to open or data to unlock. Of course, busy animations should cycle in response to the actual computations they represent. Busy animations that are simply started by a function but run independently of it are not really busy animations—they keep running even when the process they represent has hung or crashed and thereby potentially mislead users.

A common excuse for not displaying a busy indicator is that the function is supposed to execute quickly and so does not need to display one. But how quickly is “quickly”? What if the function does not always execute quickly? What if the user has a slower computer than the developer or one that is not optimally configured? What if the function tries to access data that is temporarily locked? What if the function uses network services and the network is hung or overloaded?

Software should display a busy indicator for *any* function that blocks further user actions while it is executing, even if the function normally executes quickly (e.g., in less than 0.1 second). This indicator can be very helpful to a user if for some reason the function gets bogged down or hung up. Furthermore, it harms nothing: when the function executes at normal speed, the busy indicator appears and disappears so quickly that users barely see it.

Use progress indicators

Progress indicators are better than busy indicators because they let users see how much time remains. Again, the deadline for displaying a progress indicator is 1 second.

Progress indicators can be graphical (e.g., a progress bar), textual (e.g., a count of files yet to be copied), or a combination of graphical and textual. They greatly increase the perceived responsiveness of an application even though they don’t shorten the time to complete operations.

Progress indicators should be displayed for any operation that will take longer than a few seconds. The longer the operation, the more important they are. Many noncomputer devices provide progress indicators, so we often take them for granted. Elevators that don’t show the elevator’s progress toward your floor are annoying. Most people would not like a microwave oven that didn’t show the remaining cooking time.

Here are some guidelines for designing effective progress indicators (McInerney and Li, 2002):

- Show work remaining, not work completed. Bad: “3 files copied.” Good: “3 of 4 files copied.”
- Show total progress, not progress on the current step. Bad: “5 seconds left on this step.” Good: “15 seconds left.”
- To show the percentage of an operation that is complete, start at 1%, not 0%. Users worry if the bar stays at 0% for more than 1–2 seconds.
- Similarly, display 100% only very briefly at the end of an operation. If the progress bar stays at 100% for more than 1–2 seconds, users assume it’s wrong.

- Show smooth, linear progress, not erratic bursts of progress.
- Use human-scale precision, not computer precision. Bad: “240 seconds.” Good: “About 4 minutes.”

Delays between unit tasks are less bothersome than delays within unit tasks

Unit tasks are useful not only as a way of understanding how (and why) users break down large tasks. They also provide insight into when system response delays are most and least harmful or annoying.

During execution of a unit task, users keep their goal and necessary information in working memory or within their perceptual field. After they complete one unit task, before moving onto the next one, they relax a bit and then pull the information needed for the next unit task into memory or view.

Because unit tasks are intervals during which the content of working memory and the perceptual field must remain fairly stable, unexpected system delays *during* a unit task are particularly harmful and annoying. They can cause users to lose track of some or all of what they were doing. By contrast, system delays *between* unit tasks are not as harmful or annoying even though they may slow the user’s overall work rate.

This difference between the impact of system response delays during and between unit tasks is sometimes expressed in user-interface design guidelines in terms of task closure, as in the classic user-interface design handbook *Human-Computer Interface Design Guidelines* (Brown, 1988):

A key factor determining acceptable response delays is level of closure.... A delay after completing a major unit of work may not bother a user or adversely affect performance. Delays between minor steps in a larger unit of work, however, may cause the user to forget the next planned steps. In general, actions with high levels of closure, such as saving a completed document to a file, are less sensitive to response time delays. Actions at the lowest levels of closure, such as typing a character and seeing it echoed on the display, are most sensitive to response time delays.

Bottom line: If a system has to impose delays, it should do so *between* unit tasks, not during tasks.

Display important information first

Interactive systems can appear to be operating fast by displaying important information first, then details and auxiliary information later. Don’t wait until a display is fully rendered before letting users see it. Give users something to think about and act on as soon as possible.

This approach has several benefits. It distracts users from the absence of the rest of the information and fools them into believing that the computer did what they asked quickly. Research indicates that users prefer progressive results to progress indicators (Geelhoed et al., 1995). Displaying results progressively lets users start planning their next unit task. Finally, because of the aforementioned minimum reaction time for users to respond intentionally to what they see, this approach buys at least one more second for the system to catch up before the user tries to do anything. Here are some examples:

- ***Document-editing software.*** When you open a document, the software shows the first page as soon as it has it, rather than waiting until it has loaded the entire document.
- ***Web or database search function.*** When you do a search, the application displays items as soon as it finds them while continuing to search for more.

High-resolution images sometimes render slowly, especially in web browsers. To decrease the perceived time for an image to render, the system can display the image quickly at low resolution and then rerender it at a higher resolution. Because the visual system processes images holistically, this appears faster than revealing a full-resolution image slowly from top to bottom (see [Fig. 14.3](#)). Exception: For text, rendering a page at low resolution first and then substituting a higher-resolution version is *not* recommended—it annoys users (Geelhoed et al., 1995).

Fake heavyweight computations during hand–eye coordination tasks

In interactive systems, some user actions require rapid successive adjustments—with hand–eye coordination—until the goal is achieved. Examples include scrolling through a document, moving a game character through a landscape, resizing a window, or dragging an object to a new position. If feedback lags behind user actions by more than 0.1 second, users will have trouble hitting their goal. When your system cannot update its display fast enough to meet this hand–eye coordination deadline, provide lightweight simulated feedback until the goal is clear and then apply the real operation.

Graphics editors fake feedback when they provide rubber band outlines of objects that a user is trying to move or resize. Some document-editing applications make quick-and-dirty changes to internal document data structures to represent the effect of user actions and then straighten things out later.

Work ahead

Work ahead of users when possible. Software can use periods of low load to precompute responses to high-probability requests. There will be periods of low load because users are human. Interactive software typically spends a lot of time waiting for input from the user. Don't waste that time! Use it to prepare something the user

**FIGURE 14.3**

If displaying images takes more than 2 seconds, display the whole image first at low resolution (A), not at full resolution from the top down (B).

will probably want. If the user never wants it, so what? The software did it in “free” time; it didn’t take time away from anything else. Here are some examples of using background processing to work ahead of users:

- A text search function looks for the *next* occurrence of the target word while you look at the current one. When you tell the function to find the next occurrence of the word, it already has it and so it seems very fast.
- A document viewer renders the next page while you view the current page. When you ask to see the next page, it is already ready.

Process user input according to priority, not the order in which it was received

The order in which tasks are done often matters. Blindly doing tasks in the order in which they were requested may waste time and resources or even create extra work. Interactive systems should look for opportunities to reorder tasks in their queue. Sometimes reordering tasks can make completing the entire set more efficient.

Airline personnel use nonsequential input processing when they walk up and down long check-in lines looking for people whose flights are leaving very soon so they can pull them out of line and get them checked in. In Web browsers, clicking the “Back” or “Stop” buttons or on a link *immediately* aborts the process of loading and displaying the current page. Given how long it can take to load and display a Web page, the ability to abort a page load is critical to user acceptance.

Monitor time compliance; decrease the quality of work to keep up

An interactive system can measure how well it is meeting real-time deadlines. If it is missing deadlines or determines that it is at risk of missing a pending deadline, it can adopt simpler, faster methods, usually resulting in a temporary reduction in the quality of its output. This approach must be based on *real* time, not processor cycles, so that it yields the same responsiveness on different computers.

Some interactive animation uses this technique. As described earlier, animation requires a frame rate of about 20 frames per second to be seen as smooth. In the late 1980s, researchers at Xerox Palo Alto Research Center developed a software engine for presenting interactive animations that treated the frame rate as the most important aspect of the animation (Robertson et al., 1989, 1993). If the graphics engine had trouble maintaining the minimum frame rate because the images were complex or the user was interacting with them, it simplified its rendering, sacrificing details such as text labels, three-dimensional effects, highlighting and shading, and color. The researchers considered it better to reduce an animated three-dimensional image temporarily to a line drawing than let the frame rate drop below 20/seconds.

Provide timely feedback even on the Web

Meeting the aforementioned deadlines on the Web can be difficult. However, those deadlines are psychological time constants, wired into us by evolution, that govern our perception of responsiveness. They are not arbitrary targets we can adjust at will to match the limitations of the Web or any other technology platform. Furthermore, they will not change during any timescale in which this book will be in circulation. If an interactive system does not meet those deadlines, even if it is browser based, users *will* consider its responsiveness—and therefore its quality—poor.

For example, Hoxmeier and DiCesare (2000) found that users of discretionary websites and Web apps—i.e., ones that their job does not require them to use—will not tolerate delays of more than about 10s. Longer delays make people unwilling to continue using an app or website. Echoing that deadline is Web design expert Jakob Nielsen (2010):

A 10-second delay will often make users leave a site immediately.... Even a few seconds' delay is enough to create an unpleasant user experience.... [W]ith repeated short delays, users will give up unless they're extremely committed to completing the task.

However, other researchers and design experts suggest that 10 seconds is too generous. Some found that software response times greater than 2 seconds cause users to become dissatisfied (Shneiderman, 1984; Nah, 2004). Others found that satisfaction with Web page loads decreases with increased delay, but above about 4 seconds stops decreasing, suggesting that 4 seconds is the threshold. Why the differences? Some studies tested delays in responding to clicks, while others tested Web page-load time. Some provided feedback—busy indicators, progress bars, time-remaining displays—during delays, while others did not. In addition, some studies measured satisfaction/dissatisfaction via postexperiment questionnaires, while others measured the point in time when users gave up on waiting.

Although researchers have not converged on exactly what delays users will and will not tolerate, the most important takeaway is that users won't wait forever. Developers of discretionary software apps and websites cannot simply assume that users will accept whatever response time the software delivers.

A second important takeaway is that feedback indicating that a website is working on responding increases the length of time users are willing to wait for the response (Nah, 2004).

How can designers and developers maximize responsiveness on the Web? Here are several methods:

- Minimize the number and size of images. Images for display on the Web can have lower resolution than images intended for printing.
- Provide height and width dimensions for all images so the browser can lay out the page even before it loads and displays the images. Also provide ALT text for all images so the browser can show an image's purpose before the image itself is rendered.
- Provide quick-to-display thumbnail images or overviews with ways to show details only as needed.
- When the amount of data is too large or time-consuming to display all at once, design the system to give an overview of *all* the data, and allow users to drill down into specific parts of the data to the level of detail they need.
- Style and lay out pages using cascading style sheets instead of presentational HTML, frames, or tables so browsers can display them faster.

- Use built-in browser components (e.g., error dialogue boxes) instead of constructing them in HTML, for the same reason.
- Use client-side scripts so user interactions require less Internet traffic.
- If your website or app consistently manifests noticeable delays, use performance-monitoring tools to find out where the delays are occurring—browser, Wi-Fi, user-device's Internet connection, Internet service provider, server's connection to the Internet, server-side (back-end) systems—and correct the problem if possible.

Make sure mobile apps meet human time requirements

Like Web users, mobile-phone users usually won't tolerate delays in the apps they use. Mobile design expert Craig Isakson (2013) comments: "One of the biggest downfalls of many mobile applications is their ... response time." This may seem ironic given the massive amounts of time many people waste staring at their mobile phones, but it is true. If tapping on a function button doesn't produce results within a second or two and meanwhile does not provide feedback that it detected the tap, many users will tap it again on the assumption that they missed the first time. If an app won't let users scroll smoothly through their newsfeed, won't show information they requested within a few seconds, or takes too long to upload images or videos, they will quickly abandon it.

Fortunately, the guidelines for achieving satisfactory responsiveness in mobile apps are essentially the same as those for achieving it in desktop applications and websites (see above). Most importantly, as with websites, if your app consistently manifests delays, diagnose where they are occurring and fix them.

CONCLUSION: ACHIEVING RESPONSIVENESS IS IMPORTANT

By following the responsiveness guidelines described in this chapter and the references cited here, interaction designers and developers can create apps and websites that meet human real-time deadlines and that users therefore perceive as responsive and of high quality.

However, software developers must first recognize these facts about responsiveness:

- It is of great importance to users.
- It is different from performance; responsiveness problems are not solvable merely by tuning performance or by making processors or networks faster.
- It is a *design* issue, not just an implementation issue.

History shows that faster CPUs and faster networks will not solve the problem. Today's personal computers and smartphones are as fast as supercomputers were 30 years ago, yet people still wait for their computers, tablets, and phones and grumble about a lack of responsiveness. Twenty years from now when personal computers

and electronic appliances are as powerful as *today's* most powerful supercomputers, responsiveness will still be an issue because the software then will demand much more from the machines and the networks connecting them.

For example, whereas today's applications do *spell-checking* and *grammar-checking* in the background, future versions will do Internet-based *fact-checking* in the background. Today's apps do *word completion* and *autocorrect*; tomorrow's will do *idea completion* and correction. Today's smartphones and smart speakers do speech *recognition* and *generation*; tomorrow's equivalents will be capable of speech *understanding* and *conversational dialogue*, perhaps even *argumentation*. In addition, applications 20 years from now will be based on these technologies:

- artificial intelligence, neural networks, and machine learning
- deductive reasoning
- direct brain-computer interaction
- image and video analysis and recognition
- downloading and processing terabyte-sized files
- wireless communication among dozens of household appliances
- collation of data from thousands of remote databases
- complex searches of the entire Web

The result will be systems that place a much heavier load on computers and networks than today's systems do. As computers grow more powerful, much of that power is eaten up by applications that demand ever more processing power. Therefore, despite ever-increasing performance, responsiveness will never disappear as an issue.

For common design flaws (bloopers) that hurt responsiveness, principles for designing responsive systems, and more techniques for achieving responsiveness, see Johnson (2007).

IMPORTANT TAKEAWAYS

- A user interface for an interactive system is a real-time interface—it must meet several real-time deadlines for users to perceive the system as *responsive*.
- Perceived responsiveness is an important factor in determining user satisfaction with interactive systems. It is not the same thing as performance. An interactive system can have poor responsiveness even if it has high performance, and it can have high responsiveness even if it has low performance.
- The main thing that makes a system responsive is whether it keeps users informed of its status and progress.

- The deadlines that interactive systems must meet to be perceived as responsive are determined by the *durations* of human perceptual and cognitive processes. There are dozens of time constants of human perception and cognition, but for design purposes they can be boiled down to six: 1, 10, 100 milliseconds, 1, 10, and 100 seconds.
- Responsiveness will remain an issue even as computer performance increases, because we will expect them to do more.
- Guidelines for achieving responsiveness:
 - Acknowledge user actions instantly, within 1–10 milliseconds.
 - Animate movement smoothly, with a frame rate of 10–20 frames/second.
 - Operations taking longer than 1 second should free users to do other things while waiting for a function to finish, display busy or progress indicators, and allow users to abort (cancel).
 - Avoid delays within unit tasks. Delays between unit tasks are acceptable.
 - Display important information first.
 - Work ahead of users if possible; anticipate likely requests.